Министерство образования и науки Российской Федерации
ФГБОУ ВО «Уральский государственный педагогический университет»
Институт математики, физики, информатики и технологий
Кафедра физики и математического моделирования

Разработка мобильного приложения (приложение «AvtoLife»для учета рабочего времени под платформу Android»)

Выпускная квалификационная работа

Квалификационная работа
допущена к защите
зав.кафедрой ФиММ
д.ф-м.н., профессор
Сидоров Валерий Евгеньевич

_____ _____
дата подпись

Исполнитель:
Утюмова Ксения Александровна
обучающийся БЭ-51z группы

_____
подпись

Руководитель:
Кощеева Елена Сергеевна,
к.п.н., доцент кафедры ФиММ

_____
подпись

Екатеринбург 2017

Оглавление

ВВЕДЕНИЕ

С развитием технологий, позволяющих создавать персональные мобильные устройства и различные гаджеты, корпоративный рынок получил мощнейший стимул к развитию. Полюбившиеся миллионам современные смартфоны люди носят с собой всегда и везде. Телефоны играют важную роль в повседневной работе: с их помощью читают файлы, просматривают почту, печатают документы при помощи принтера. В связи с изменениями, на рынке постепенно сформировался отдельный сегмент – мобильные приложения. Устройство смартфона является не таким сложным, оно представляет собой несколько блоков – операционной и встроенной памяти, процессора, необходимого для различных вычислений, хранилища данных и отвечающего за связь радиомодуля, свою очередь образованного из передатчика и приемника. От простого мобильного телефона смартфоны отличаются главным образом наличием операционной системы. Лидирующие позиции на сегодняшний день занимают платформы Android и iPhone. Но эти платформы могут работать полноценно только при одном условии - если была для них осуществлена разработка мобильных приложений.

Цель работы: разработка мобильного приложения учета рабочего времени для смартфонов под управлением операционной системы Android. Для возможности работы мобильного приложения на старых версиях платформы Android было решено поддерживать устаревшие версии платформы, начиная с версии 5.0. Целью этого мобильного приложения является повышение удобства записи учета рабочего времени или иных заметок.

Задача работы: предоставление зарегистрированным пользователям по управлению таймером перед началом рабочего дня, посредством включения кнопки.

Мобильное приложение может работать без соединения с сетью Интернет. Целевой аудиторией мобильного приложения являются потенциальные пользователи, которым нужно удобное в использовании средство для управления временем в дороге. Аналогичных сервисов в магазине Google Play не большое количество, есть приложения общие по возможностям, а также есть отличительные особенности. Но за функционалом скрывается простая потребность пользователей подобных приложений в удобном и быстром способе учета рабочего времени, записи времени в дороге, для того чтобы не держать информацию в голове. В погоне за пользователями создатели приложений заинтересовывают новым функционалом, которого нет у аналогов. В данном приложении сохранена простота и конкретика решаемой задачи – возможность быстро включить таймер в телефоне, для начала отчета рабочего дня. Для того чтобы взаимодействовать с приложением не нужно делать много действий и перемещаться по огромному количеству меню, которых в изобилии в аналогичных приложениях.

Задачи работы:

1. Изучить и освоить знания в разработке приложений для мобильных устройств под Android.
2. Ознакомиться с приложениями и особенностями платформы.
3. Показать значимость разработки приложения под Android на Java.
4. Разработать мобильное приложение для учета времени.

Практическая значимость исследования заключается в том, что работа может быть использована в профессиональной деятельности разработчика мобильных приложений. Кроме того, материал работы может использоваться как портфолио.

ГЛАВА I Современные средства реализации мобильных приложений

1.1 Основные характеристики операционной системы ANDROID

Знакомство с разработкой в области приложений под операционную систему Android, предоставляет отличную возможность по написанию программ для мобильных устройств. Android это набор программ с открытым исходным кодом, который включает операционную систему, подпрограммное обеспечение и ключевые мобильные приложения вместе с библиотеками API, предназначенными для написания новых программ, определяющих визуальное представление и функционал мобильных устройств. Самые разнообразные мобильные устройства со временем снабжались такими мощными инструментами, как камера, медиа-плеер, навигатор, сенсорный дисплей. С внедрением новых технологий мобильный телефон превратился в нечто большее, чем просто устройство для звонков.

При этом программная платформа и среда разработки отставали от бешеного темпа развития технологий. До недавнего времени мобильные телефоны работали под управлением закрытых платформ, построенных на основе запатентованных операционных систем, для чего требовались запатентованные инструменты разработки. Сами же телефоны функционировали с оригинальным программным обеспечением гораздо лучше, чем со сторонним. Это создавало искусственные препятствия для программистов, которые рассчитывали на использование более мощного аппаратного обеспечения мобильных устройств. В случае с Android встроенное ПО написано на том же самом API, что и программы сторонних разработчиков, при этом время для исполнения и тех, и других одинаково. Данное API позволяет получить доступ к сенсорному управлению, навигационным сервисам, фоновым и картографическим процессам, реляционным базам данных, двухмерной и трехмерной графике, к функциям видеозаписи, межпрограммного взаимодействия. В распоряжении

разработчика приложений для Android достаточно мощное API и качественная справочная документация. Он может стать членом огромного сообщества, ему не нужно тратиться на программное обеспечение или рекламу своего продукта. С ростом популярности мобильных устройств открываются великолепные перспективы разрабатывать инновационные мобильные приложения, причем с любым опытом программирования. [2]

Android — одна из операционных систем нового поколения, созданных для работы с аппаратным обеспечением современных мобильных устройств. На сегодняшний день Windows Mobile и Apple iPhone предлагают достаточно мощные и более простые в использовании среды разработки мобильных приложений. Однако в отличие от Android это запатентованные операционные системы, в которых в определенных случаях приоритет отдается встроенному ПО, а не приложениям сторонних программистов. Кроме того, эти операционные системы ограничивают возможности взаимодействия приложений с данными телефона, а также ограничивают или контролируют процесс распространения сторонних приложений, созданных для данных платформ. Android дает новые возможности для мобильных приложений, предлагая открытую среду разработки, построенную на открытом ядре Linux. У всех приложений есть доступ к аппаратным средствам устройства, для чего используются специальные серии API-библиотек. Кроме того, здесь включена полная и контролируемая поддержка взаимодействия приложений. На платформе Android все программы имеют одинаковый статус. Сторонние приложения написаны на том же API, что и встроенное ПО, при этом во всех программах одинаковое время исполнения. Пользователи могут удалять или заменять встроенные ПО на альтернативные сторонние разработки, например, игровое приложение или рабочий стол.

История Android началась в далеком 2002 году, когда корпорация Google заинтересовались наработками Энди Рубина и решила сделать из этого большой проект. И в 2007 году Google решил организовать большой альянс разработчиков мобильных устройств с целью продвигать Android, как

операционную систему для телефонов. Как мы видим на сегодняшний день это у них отлично получилось.

Android — операционная система для смартфонов, планшетов, электронных книг, цифровых проигрывателей, наручных часов, игровых приставок, нетбуков, смартбуков и других устройств. Основана на ядре Linux и собственной реализации виртуальной машины Java от Google.

Архитектуру Android принято делить на четыре уровня:

уровень ядра;

уровень библиотек и среды выполнения;

уровень каркаса приложений;

уровень приложений.

На рис. 1.1 показаны основные компоненты операционной системы Android и их взаимодействие между собой.

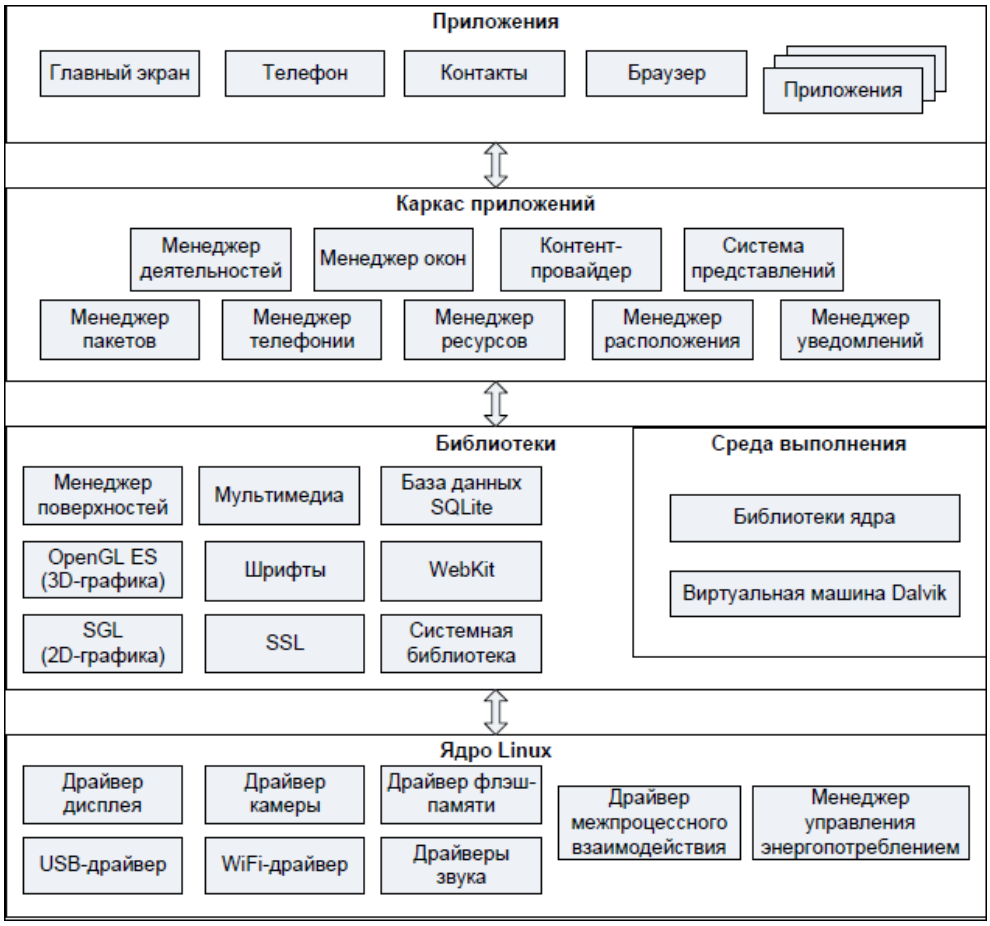


Рис. 1.1. Основные компоненты системы Android

По итогам 2014 года Android занимает лидирующую позицию на рынке операционных систем, в 86 % смартфонов, проданных во втором квартале 2014 года, была установлена операционная система Android. На конференции для разработчиков в мае 2017 года Google объявила, что за всю историю Android было активировано более 2 млрд Android-устройств. Прогнозы относительно данной платформы положительные. Немаловажными аспектами роста динамики явились такие характеристики как, открытость системы, возможность вносить изменения в основные приложения, возможность быстрой и легкой разработки. Поэтому выбор разработки приложения для Android является очень актуальным.

Разработчику работающему с платформой Android дается возможность писать код на Java абстрагируясь от ядра. У данной операционной системы имеются такие плюсы как: фреймворк, имеющий широкий набор API для созданий разнообразных видов приложений и дающий возможность повторного использования и замены компонентов, предлагаемые платформой и другими приложениями. А также наличие виртуальной машины Dalvik, обеспечивающий запуск приложений. Кроме того инструментами Android являются база данных SQLite, 2D и 3D графика, Media Player, коммуникации, протоколы обмена и различные библиотеки. [4]

Необходимо установить IDE приложение, помогающее программистам в написании кода. Это приложение предоставляет собой сжатый набор инструментов вроде отладчиков, компиляторов и многого другого. Такие интегрированные среды разработки используются сезонными разработчиками и новичками, имеющие желание создать приложение

Android SDK включает в себя необходимые инструменты, чтобы помочь Android-разработчикам в совершении первых шагов: различные API (прикладной программный интерфейс), разработанные Google для управления функциями устройства и интеграции сервисов, полнофункциональный эмулятор для тестирования приложений, а также все

необходимые текстовые материалы, чтобы дать начало для программирования под Android.

При завершении скачивания исполнительного файла, требуется запустить установку. Android Software Development Kit состоит из документации, утилит, широкого набора инструментов и различных примеров. SDK включает в себя, отладчик, профиль памяти и производительности, необходимый для обнаружения утечки памяти и поиска неэффективных кодов, эмулятор устройства, утилиты необходимые для связи с устройствами и создание пакетов.

Если создание программ для Android является новым делом, важно усвоить следующие основные концепции, касающиеся платформы приложений Android. Приложения для Android строятся из отдельных компонентов, которые можно вызывать независимо друг от друга. Например, отдельная операция предоставляет один экран для пользовательского интерфейса, а служба независимо выполняет работу в фоновом режиме.

С помощью объекта Intent из одного компонента можно запустить другой компонент. Можно даже запустить компонент из другого приложения, скажем, операцию из картографического приложения, чтобы показать адрес. Эта модель формирует несколько точек входа для одного приложения, и при этом пользователь может выбрать любое приложение для выполнения по умолчанию того или иного действия, которое могут вызывать другие приложения.

Приложения адаптируются к различным устройствам. Android предоставляет адаптивную платформу приложений, которая позволяет обеспечивать уникальные ресурсы для различных конфигураций устройств. Например, можно создать разные файлы XML макета для экранов разных размеров, а система будет определять, какой макет использовать, с учетом размера экрана данного устройства.

Если каким-либо функциям приложения требуется определенное оборудование, например камера, можно запрашивать его наличие в

устройстве во время выполнения. При необходимости также можно объявлять функции, которые требуются приложению, с тем чтобы такие магазины приложений, как Google Play, не позволяли устанавливать приложения на устройствах, в которых этой функции нет.

Каждое приложение, установленное на устройстве, работает в собственной "песочнице" (изолированной программной среде), операционная система Android представляет собой многопользовательскую систему Linux, в которой приложение является отдельным пользователем; по умолчанию система назначает каждому приложению уникальный идентификатор пользователя Linux (этот идентификатор используется только системой и неизвестен приложению); система устанавливает полномочия для всех файлов в приложении, для того чтобы доступ к ним был разрешен только пользователю с идентификатором, назначенным этому приложению. У каждого процесса имеется собственная виртуальная машина (ВМ), так что код приложения выполняется изолированно от других приложений; по умолчанию каждое приложение выполняется в собственном процессе Linux. Android запускает процесс, когда требуется выполнить какой-либо компонент приложения, а затем завершает процесс, когда он больше не нужен либо когда системе требуется освободить память для других приложений.

Таким образом система Android реализует принцип предоставления минимальных прав. То есть каждое приложение по умолчанию имеет доступ только к тем компонентам, которые ему необходимы для работы, и ни к каким другим. Благодаря этому формируется исключительно безопасная среда, в которой приложение не имеет доступа к недозволенным областям системы.

Однако у приложения есть варианты предоставления своих данных другим приложениям и доступа к системным службам. Двум приложениям можно назначить один идентификатор пользователя Linux. В этом случае каждый из них сможет обращаться к файлам другого приложения. Для

экономии ресурсов системы также можно сделать так, чтобы приложения с одинаковым идентификатором пользователя выполнялись в одном процессе Linux и использовали одну ВМ (приложения также должны быть подписаны одним сертификатом); приложение может запросить разрешение на доступ к данным устройства, например к контактам пользователя, SMS-сообщениям, подключаемой карте памяти (SD-карте), камере, Bluetooth и др. Все разрешения должны предоставляться приложению при его установке.

Это основные сведения о том, каким образом приложение Android существует в системе. В остальной части раскрываются следующие темы:

-базовые компоненты, которые определяют приложение;

-файл манифеста, в котором объявляются компоненты и функции устройства, необходимые для приложения;

-ресурсы, которые существуют отдельно от кода приложения и позволяют приложению адаптировать свою работу к устройствам с различными конфигурациями.

Компоненты приложения являются кирпичиками, из которых состоит приложение для Android. Каждый компонент представляет собой отдельную точку, через которую система может войти в приложение. Не все компоненты являются точками входа для пользователя, а некоторые из них зависят друг от друга. При этом каждый компонент является самостоятельной структурной единицей и играет определенную роль — каждый из них представляет собой уникальный элемент структуры, который определяет работу приложения в целом.[25]

Компоненты приложения можно отнести к одному из четырех типов. Компоненты каждого типа предназначены для определенной цели, они имеют собственный жизненный цикл, который определяет способ создания и прекращения существования компонента.

Четыре типа компонентов:

-операция (Activity) представляет собой один экран с пользовательским интерфейсом. Например, в приложении для работы с электронной почтой

одна операция может служить для отображения списка новых сообщений, другая – для составления сообщения и третья операция – для чтения сообщений. Несмотря на то что операции совместно формируют связное взаимодействие пользователя с приложением по работе с электронной почтой, каждая из них не зависит от других операций. Любые из этих операций могут быть запущены другим приложением (если это позволяет приложение по работе с электронной почтой). Например, приложение для камеры может запустить операцию в приложении по работе с электронной почтой, которая составляет новое сообщение, чтобы пользователь мог отослать фотографию;

- служба (Service) представляет собой компонент, который работает в фоновом режиме и выполняет длительные операции, связанные с работой удаленных процессов. Служба не имеет пользовательского интерфейса. Например, она может воспроизводить музыку в фоновом режиме, пока пользователь работает в другом приложении, или же она может получать данные по сети, не блокируя взаимодействие пользователя с операцией. Служба может быть запущена другим компонентом, который затем будут взаимодействовать с ней, – например операцией;

- поставщик контента (Content provider) управляет общим набором данных приложения. Данные можно хранить в файловой системе, базе данных SQLite, в Интернете или любом другом постоянном месте хранения, к которому у вашего приложения имеется доступ. Посредством поставщика контента другие приложения могут запрашивать или даже изменять данные (если поставщик контента позволяет делать это). Например, в системе Android есть поставщик контента, который управляет информацией контактов пользователя. Любое приложение, получившее соответствующие разрешения, может запросить часть этого поставщика контента (например ContactsContract.Data), для чтения и записи сведений об определенном человеке;

- приемники широковещательных сообщений

Приемник широковещательных сообщений (Broadcast receiver) представляет собой компонент, который реагирует на объявления распространяемые по всей системе. Многие из этих объявлений рассылает система — например объявление о том, что экран выключился, аккумулятор разряжен или был сделан фотоснимок. Объявления также могут рассылаться приложениями, — например, чтобы сообщить другим приложениям о том, что какие-то данные были загружены на устройство и теперь готовы для использования. Несмотря на то что приемники широковещательных сообщений не имеют пользовательского интерфейса, они могут создавать уведомления в строке состояния, чтобы предупредить пользователя о событии "рассылка объявления". Однако чаще всего они являются просто "шлюзом" для других компонентов и предназначены для выполнения минимального объема работы. Например, они могут инициировать выполнение службой определенных действий при возникновении события.[6]

Уникальной особенностью системы Android является то, что любое приложение может запустить компонент другого приложения. Например, предоставить пользователю возможность фотографировать, используя камеру устройства, то, поскольку наверняка имеется другое приложение, которое может выполнить это действие, вместо того чтобы разработать операцию фотографирования в своем приложении, вы можете вызвать такое приложение. Вам не нужно внедрять код из приложения для камеры или даже устанавливать на него ссылку. Вместо этого вы можете просто запустить операцию фотографирования из приложения для камеры. По завершении этой операции фотография будет возвращена в ваше приложение, и ее можно будет использовать. Для пользователя это будет выглядеть как одно приложение.

Когда система запускает компонент, она запускает процесс для этого приложения (если он еще не был запущен) и создает экземпляры классов, которые требуются этому компоненту. Например, если ваше приложение запустит операцию фотографирования в приложении для камеры, эта

операция будет выполняться в процессе, который относится к этому стороннему приложению, а не в процессе вашего приложения. Поэтому, в отличие от приложений для большинства других систем, в приложениях для Android отсутствует единая точка входа (например, в них нет функции main()).

Поскольку система выполняет каждое приложение в отдельном процессе с такими правами доступа к файлам, которые ограничивают доступ в другие приложения, ваше приложение не может напрямую вызвать компонент из другого приложения. Это может сделать сама система Android. Поэтому, чтобы вызвать компонент в другом приложении, необходимо сообщить системе о своем намерении (Intent) запустить определенный компонент. После этого система активирует для вас этот компонент.

Активация компонентов

Компоненты трех из четырех возможных типов — операции, службы и приемники широковещательных сообщений — активируются асинхронным сообщением, которое называется Intent (намерение). Объекты Intent связывают друг с другом отдельные компоненты во время выполнения, будь то это компоненты вашего или стороннего приложения (эти объекты Intent можно представить себе в виде мессенджеров, которые посылают другим компонентам запрос на выполнение действий).

Объект Intent создается с помощью объекта Intent, который описывает запрос на активацию либо конкретного компонента, либо компонента конкретного типа — соответственно, намерение Intent может быть явным или неявным.

Для операций и служб Объект Intent определяет действие, которое требуется выполнить (например, просмотреть (view) или отправить (send) что-то), а также может указывать URI (Uniform Resource Identifier – унифицированный идентификатор ресурса) данных, с которыми это действие нужно выполнить (помимо прочих сведений, которые нужно знать запускаемому компоненту). Например, объект Intent может передавать запрос

на выполнение операции "показать изображение" или "открыть веб-страницу". В некоторых ситуациях операцию можно запустить, чтобы получить результат. В этом случае операция возвращает результат также в виде объекта Intent (например, можно отправить сообщение Intent, чтобы дать пользователю возможность выбрать контакт и вернуть его вам — в ответном сообщении Intent будет содержаться URI, указывающий на выбранный контакт).

Для приемников широковещательных сообщений Intent просто определяет передаваемое объявление (например, широковещательное сообщение о низком уровне заряда аккумулятора содержит только строку "аккумулятор разряжен").

Компоненты четвертого типа – поставщики контента – сообщениями Intent не активируются. Они активируются по запросу от ContentResolver. Процедура определения контента (content resolver) обрабатывает все прямые транзакции с поставщиком контента, с тем чтобы этого не пришлось делать компоненту, который выполняет транзакции с поставщиком. Вместо этого он вызывает методы для объекта ContentResolver. Это формирует слой, абстрагирующий (в целях безопасности) поставщика контента от компонента, запрашивающего информацию.

Для запуска компонента приложения системе Android необходимо знать, что компонент существует. Для этого она читает файл AndroidManifest.xml приложения (файл манифеста). В этом файле, который должен находиться в корневой папке приложения, должны быть объявлены все компоненты приложения.

Помимо объявления компонентов приложения, манифест служит и для других целей, среди которых:

-указание всех полномочий пользователя, которые требуются приложению, например разрешения на доступ в Интернет или на чтение контактов пользователя;

-объявление минимального уровня API, требуемого приложению, с учетом того, какие API-интерфейсы оно использует;

-объявление аппаратных и программных функций, которые нужны приложению или используются им, например камеры, службы Bluetooth или сенсорного экрана;

-указание библиотек API, с которыми необходимо связать приложение (отличные от API-интерфейсов платформы Android), например библиотеки Google Maps.

Объявление компонентов

Основная задача манифеста – это информировать систему о компонентах приложения.

Объявление требований приложения

Существует огромное количество устройств, работающих под управлением Android, и не все они имеют одинаковые функциональные возможности. Чтобы ваше приложение не могло быть установлено на устройствах, в которых отсутствуют функции, необходимые приложению, важно четко определить профиль для типов устройств, поддерживаемых вашим приложением, указав требования к аппаратному и программному обеспечению в файле манифеста. Эти объявления по большей части носят информационный характер, система их не читает. Однако их читают внешние службы, например Google Play, с целью обеспечения фильтрации для пользователей, которые ищут приложения для своих устройств.[28]

Java Development Kit –это кроссплатформенный интерактивный пакет созданный для разработчиков работающих на языке Java.[19] Комплект состоит из нескольких компонентов таких как компилятор java, стандартные библиотеки, примеры и шаблоны, также утилиты необходимые для работы.

Система Android предоставляет разностороннюю платформу приложений, на основе которой можно создавать инновационные приложения и игры для мобильных устройств в среде языка Java.

Наибольшей популярностью пользуются приложения для Android, разработанные на программной платформе Java.

Использование этого языка дает следующие преимущества:

богатый функционал;

кроссплатформенность;

простое обновление программного обеспечения;

создание гибридных приложений.

Java — полностью объектно-ориентированный язык, отвечая потребностям возникшей интерактивной среды, Java предоставляет средства, упрощающие создание прикладных программ с распределенной архитектурой.

Первоначально он назывался Oak, но в 1995 году, когда за его продвижение на рынке взялись маркетологи, он был переименован в Java. Как это ни удивительно, на первых порах сами разработчики языка не ставили перед собой задач разработки интернет - приложений. Их целью было создание платформенно - независимого языка, на котором можно было бы писать встраиваемое программное обеспечение для различной бытовой аппаратуры с микропроцессорным управлением, в том числе тостеров, микроволновых печей и пультов дистанционного управления. На сегодняшний день создание программного обеспечения представляет собой чрезвычайно тяжелое занятие. Трудности связаны с разнообразием архитектур машин, операционных систем и графических оболочек. Кроме того, приложения должны работать в распределенных системах.

Стремительный рост технологий, связанных с Интернетом и "электронной коммерцией", дополнительно усложняют эту задачу. Модный ныне объектно-ориентированный подход сам по себе не решает этих проблем, более того, часто привносит новые.

Именно система программирования на основе языка Java(TM) обладает следующими характеристиками:

-язык программирования объектно-ориентирован, в то же время довольно прост для освоения

-цикл разработки приложений сокращен за счет того, что система построена на основе интерпретатора

-приложение получается автоматически переносимым между множеством платформ и операционных систем

-за счет встроенной системы сборки мусора программист освобождается от необходимости явного управления памятью

-в интерактивном графическом приложении удается достичь высокой производительности (быстрого отклика на ввод пользователя) за счет встроенной в систему многопотоковости

-приложение легко сопровождается и модифицируется, т.к. модули могут быть загружены с сети

-в приложения встроена система безопасности, не допускающая незаконного доступа и проникновения вирусов.

Система Java создана на основе «простого» языка программирования, техника использования которого близка к общепринятой, и обучение которому не требует значительных усилий. Java как язык программирования является объектно-ориентированной с момента основания. Кроме того программист с самого начала обеспечивается набором «стандартных» библиотек, обеспечивающих функциональность от стандартного ввода/вывода и сетевых протоколов до графических пользовательских интерфейсов. Эти библиотеки легко могут быть расширены.

Несмотря на то, что язык C++ был отвергнут, синтаксис языка Java максимально приближен к синтаксису C++. Это делает язык знакомым широкому кругу программистов. В то же время из языка были удалены многие свойства, которые делают C++ излишне сложным для пользования, не являясь абсолютно необходимыми. В результате язык Java получился более простым и органичным, чем C++.

Надежность и безопасность

Java существенно облегчает создание надежного программного обеспечения. Кроме исчерпывающей проверки на этапе компиляции, система предусматривается анализ на этапе выполнения. Сам язык спроектирован так, чтобы вырабатывать у программиста привычку писать "правильно". Модель работы с памятью, в которой исключено использование указателей, делает невозможными целый класс ошибок, характерных для C и C++. В силу того, что Java предназначена для работы в распределенной среде, безопасность становится чрезвычайно важной проблемой. Требования безопасности определяют многие черты как языка, так и реализации всей системы.

Независимость от архитектуры и переносимость.

Компилятор Java производит байт-коды, т.е. модули приложения имеют архитектурно-независимый формат, который может быть проинтерпретирован на множестве разнообразных платформ. Это уже не исходные тексты, но еще не платформно-зависимые машинные коды.

Следующий шаг - "замораживание" стандарта на формат основных встроенных типов данных. Программа, созданная на одной платформе, работает на всех остальных.

Этот стандарт фиксирован в документе, описывающем Java Virtual Machine. Стандарт может быть реализован на любой аппаратно-программной платформе, поддерживающей многопотоковость.

Производительность

Схема работы системы и набор байт-кодов виртуальной машины Java таковы, что позволяют достичь высокой производительности на этапе выполнения программы:

-анализ кодов на соблюдение правил безопасности производится один раз до запуска кодов на выполнение, в момент выполнения таких проверок уже не нужно, и коды выполняются максимально эффективно

-работа с базовыми типами максимально эффективна, для операций с ними зарезервированы специальные байт-коды

-методы в классах не обязательно связываются динамически

-автоматический сборщик мусора работает отдельным фоновым потоком, не замедляя основную работу программы, но в то же время обеспечивая своевременный возврат свободной памяти в систему

-стандарт предусматривает возможность написания критических по производительности участков программы в машинных кодах

Интерпретируемый, многопотоковый и динамический.

Интерпретируемая природа языка позволяет сделать фазу линкования простой, инкрементальной и, следовательно, быстрой. Это резко сокращает цикл разработки и тестирования программных фрагментов. Многопотоковость позволяет выполнять в рамках одного приложения несколько задач одновременно. Это становится особенно актуально в современных распределенных приложениях, когда процессы сетевого обмена могут идти одновременно и асинхронно. При этом программа продолжает реагировать на ввод информации пользователем без неприятных задержек.

Многопотоковость поддерживается на уровне языка - часть примитивов синхронизации встроена в систему реального времени, а библиотека содержит базовый класс Thread. К тому же системные библиотеки написаны thread-safe, т.е. все они могут быть использованы в многопотоковых приложениях. Система обеспечивает динамическую сборку программы. Классы подгружаются по мере необходимости, причем загружены они могут быть с любой точки сети, что позволяет сделать внесение изменений в приложения прозрачным для пользователя. Пользователь может быть уверен, что всегда работает со свежей версией приложения.[7]

Полная система Java включает в себя готовый набор библиотек, который можно разбить на следующие пакеты:

-java.lang - базовый набор типов, отраженных в самом языке. Этот пакет обязательно входит в состав любого приложения. Содержит описания классов Object и Class, а также поддержку многопотоковости,

исключительных ситуаций, оболочку для базовых типов, а также некоторые фундаментальные классы.

-java.io - потоки и файлы произвольного доступа. Аналог библиотеки стандартного ввода-вывода системы UNIX. Поддержка сетевого доступа (sockets, telnet, URL) содержится в пакете java.net.

-java.util - классы-контейнеры (Dictionary, HashTable, Stack) и некоторые другие утилиты. Кодирование и декодирование. Классы Date и Time.

-java.awt - Abstract Windowing Toolkit, архитектурно-независимый оконный интерфейс, позволяющий запускать интерактивные оконные Java-приложения на любой платформе. Содержит базовые компоненты интерфейса, такие как события, цвета, фонты, а также основные оконные элементы - кнопки, scrollbars и т.д..

Каждая из перечисленных характеристик по отдельности может быть найдена в уже существующих программных пакетах. Новым является соединение их в стройную непротиворечивую систему, которая должна стать всеобщим стандартом.

Основные свойства языка программирования Java

Встроенные (примитивные) типы данных

В языке Java существует набор встроенных типов данных, которые не являются объектами. Набор их сходен с набором базовых типов C++ за некоторыми исключениями.

Отличаются от C++ как синтаксисом, так и представлением. Тип character есть 16-разрядное число без знака (диапазон 0-65,535). Кодировка соответствует стандарту Unicode. В силу того, что эта кодировка в идеале должна охватывать все существующие в мире языки, это представление должно облегчить локализацию приложений.

Boolean этот тип данных не выделен в C++, однако неявно присутствует практически во всех программах. В Java тип boolean, может

принимать значения true и false и не может (в отличие от C++) быть преобразован в другой тип.

Добавлен новый оператор логического сдвига вправо (т.к. нет беззнаковых целых чисел). Встроенная операция слияния строк (оператор +).

Массивы, в отличие от C++ массивы в Java являются полноценными объектами с определенным runtime представлением

Система Java создавалась объектно-ориентированной с самого начала. Объектно-ориентированная парадигма наиболее удобна при создании программного обеспечения типа клиент-сервер, а также для организации распределенных вычислений. Одна из черт, присущих объектам, заключается в том, что объекты обычно переживают процедуру, их создающую. Они затем могут перемещаться по сети и храниться в базах данных.

Основные требования к объектно-ориентированной системе

-инкапсуляция - сокрытие реализации за абстрактным интерфейсом

-полиморфизм - одно и то же сообщение, посланное различным объектам, приводит к выполнению разных операций

-наследование - новые классы могут наследовать данные и функциональность уже существующих классов

-динамическое связывание - новые классы могут появляться в системе откуда угодно, в том числе и из сети. Необходимо иметь возможность динамически включать их в систему.

Классы

Класс есть языковая конструкция, определяющая поля данных объектов данного класса (instance variables) и их поведение (methods). Практически класс в Java сам по себе не является объектом. Это лишь шаблон, который определяет, из каких частей будет состоять объект, созданный с помощью этого класса, и как он будет себя вести. Освещены следующие стороны Java как объектно-ориентированного языка программирования.

-Классы определяют шаблон, по которому создаются конкретные объекты

-Поля данных объекта определяют состояние объекта

-Объекты обмениваются сообщениями между собой. Получение сообщения приводит к вызову одного из методов

-Методы определяют поведение объекта данного класса. Методы для разных классов могут иметь одно и то же имя, но различное содержание.

Система Java предназначена для создания программного обеспечения, которое должно быть интеллектуальным, предельно надежным и безопасным по множеству параметров. Особое внимание уделяется как ранней диагностике возможных проблем, так и поздней, во время выполнения кодов.

Система Java достаточно безопасна, чтобы жить в сетевом окружении. Нейтральность к архитектуре и переносимость делают ее достаточно привлекательной для создания распределенных по сети приложений.

1.2 Обзор приложений для учета рабочего времени водителя

Аналогами мобильного приложения “AvtoLife” являются проекты вроде “AntiBAG Tahograf” и “ Intempus Timeregistrering”.

Приложение “AntiBAG Tahograf” :

«В приложении есть три активных области.

Первая - это напоминание ежедневного ввода страны начала и конца работы. Ввели в тахографе букву и нажали соответствующую кнопку в приложении. Если забыли сокращенную букву страны для тахографа, то в приложении AntiBAG Tahograf смахните справа на лево и появиться список стран и сокращенных обозначений этих стран. Нажатие в нижнюю пустую область вернет вас обратно к программе. В конце рабочего дня ввели в

тахографе букву страны окончания работы и подтвердили это действие в приложении.

Вторая область это все необходимые таймеры для работы. Удержание кнопки любого таймера в течении одной секунды запускает таймер или ставит его на паузу, если таймер уже был запущен. Удержание кнопки в течении 3 секунд сбрасывает таймер в первоначальное положение. Все таймеры работают независимо друг от друга.

Перед началом работы включите таймер общего времени 13/15 часов(ALL) и вы всегда будете знать сколько у Вас осталось рабочего времени. За 30 минут приложение сообщит включением экрана и звуком о завершении рабочего дня. Если превысили положенное время, то приложение покажет на сколько минут.

Таймер 45 минут проинформирует Вас об окончании паузы, когда вы находитесь вне автомобиля или заняты своими делами. Даст сигнал на 15 и 45 минутах. Если пауза нажата до 15 минут, то отдых не засчитается. Если 15 минут прошли, а 45 минут еще не наступили, то засчитается только 15 минут, а следующую паузу таймер будет считать 30 минут.

Таймер 9/11 часов рассчитает вам междневную паузу. Вы всегда будете видеть сколько вам осталось стоять до пределов 9 и 11 часов. Таймер 24/45 часов рассчитает вам еженедельную паузу, Вы всегда будете видеть сколько вам осталось стоять до пределов 24 и 45 часов, если пауза сокращённая то приложение точно покажет время компенсации.

Таймер 144 часа (W - W) очень важный таймер. Его нужно включить тогда когда начали свой первый рабочий день, или когда вы закончили межнедельную паузу и начинается новая рабочая неделя. Вы всегда будете видеть сколько осталось времени до наступления начала следующей межнедельной паузы. 144 часа это максимально допустимый промежуток между двумя межнедельными паузами. Таймер сообщит вам за 30 минут о необходимости встать на очередной межнедельный отдых. Он должен работать всю рабочую неделю.

Третья область это рабочий календарь. Сделали 15 или 9 часов, нажали соответствующую кнопку. Так вы всегда видите всю нужную информацию за рабочую неделю. Если между двумя межнедельными отдыхами уже были три увеличенных рабочих дня(15 часов) или три сокращённый ежедневных отдыха(9 часов), то таймер общего рабочего времени(ALL) будет считать только до 13 часов.

Если вы проехали больше 9 часов, то нажмите кнопку 10. В календарной неделе можно 2 раза управлять машиной 10 часов. Индикация в рабочем календаре не позволит превысить этот недельный лимит. Индикаторы 10 часовой езды сбросятся автоматически в понедельник 00:00.

Индикатор-кнопка 24/45 напомнит какой отдых был у Вас предыдущий раз, сокращённый или полный. Это не позволит вам сделать подряд два сокращенных межнедельных отдыха. Переключать этот индикатор нужно в конце межнедельной паузы. Переключение этого индикатора сбрасывает кнопки 15 и 9 в календаре и с началом новой рабочей недели вы можете опять брать 15 часовые рабочие дни и сокращенные 9 часовые паузы.

При движении пальцев по экрану по вертикали включается дневной/ночной режим отображения.

Программа полностью защищена от любых отключений, таких как принудительное завершение или разрядка аккумулятора.

Когда вы установили АнтиБАГ Тахограф и сделали первый запуск, пройдет проверка лицензии и интернет можно отключать.

По свайпу слева на право появился экран выбора работы в одиночном экипаже или в паре.»

Продавец: ZlojDalnoboy

Количество установок: 1000-5000

Обновлено: 19 июня 2017 года

Стоимость: 690 рублей

Мобильное приложение изображено на рисунках 1.2-1.5

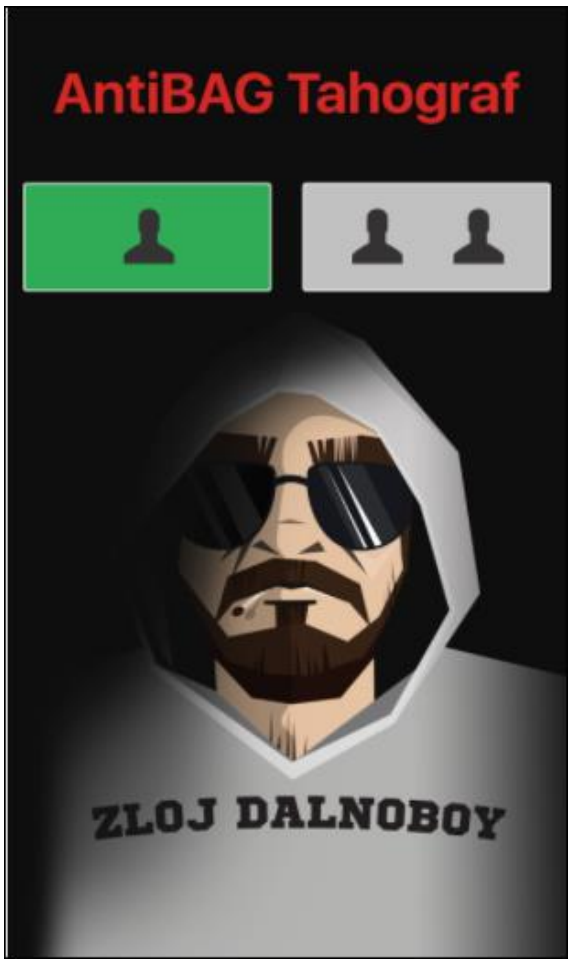


Рис. 1.2. Начальный экран

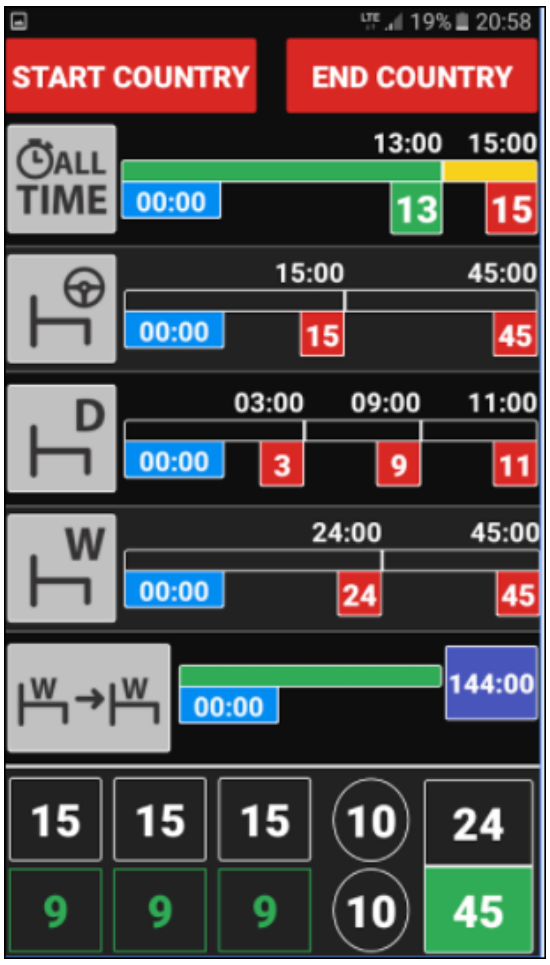


Рис. 1.3. Вторая область

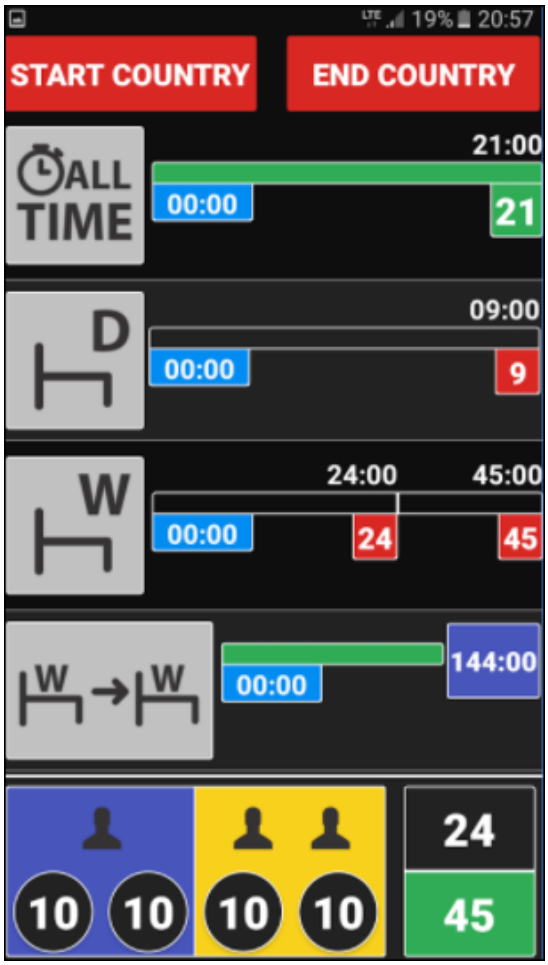


Рис. 1.4. Третья область

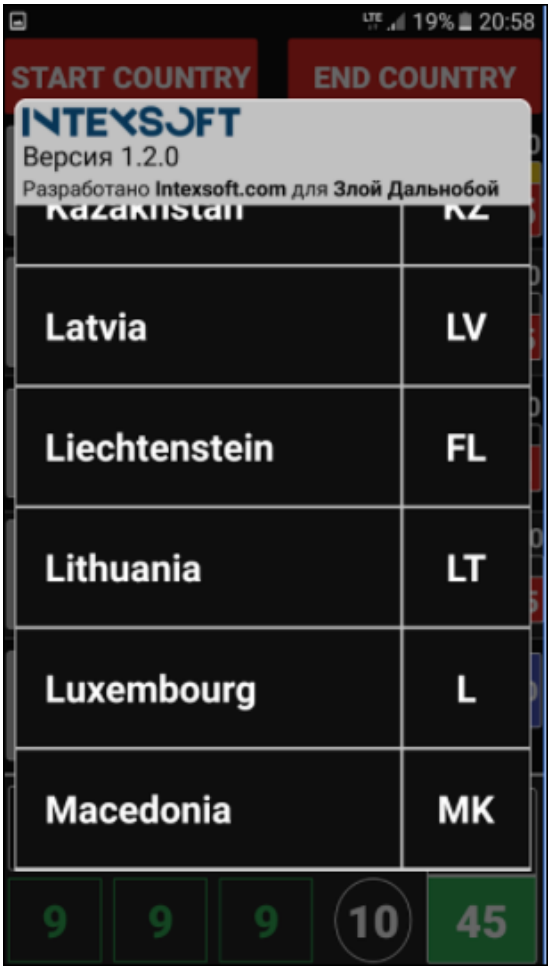


Рис. 1.5. Ввод страны

Приложение Intempus Timeregistrering":

«Intempus - приложение для регистрации таймера, которое может записывать всю вашу работу - и немного больше.

С помощью приложения Intempus вы можете зарегистрироваться:

- Материалы / Товары
- Вождение
- паузы
- Время работы машины
- Сборы
- сверхурочное время
- Дополнение

И почему просто Intempus App?

- Работает офлайн

- Удобный дизайн

- Обзор временной регистрации дел и сотрудников

- Создание дел и клиентов на ходу

- Простое управление проектами

- Бесплатная пробная версия на 30 дней - совершенно необязательна!

Администрация Intempus, веб-система администрирования, предоставляет администрации Intempus обзор времени регистрации ваших сотрудников. Обзор позволяет легко извлекать файлы Excel.

На самом деле, Команда Intempus

Оригинальный язык Датский, перевод выполнен с помощью Google переводчика.

Продавец: Intempus ApS

Количество установок: 10000-50000

Обновлено: 29 октября 2017 года

Стоимость: бесплатно

Мобильное приложение изображено на рисунках 1.6.-1.11.

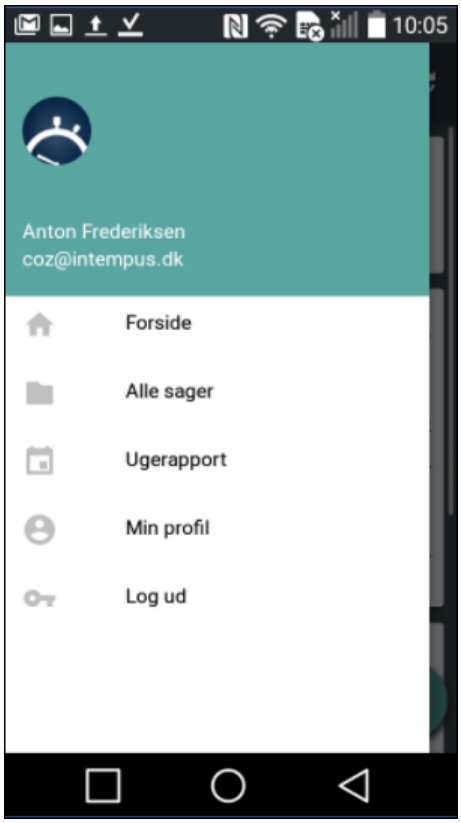


Рис. 1.6. Экран профиля

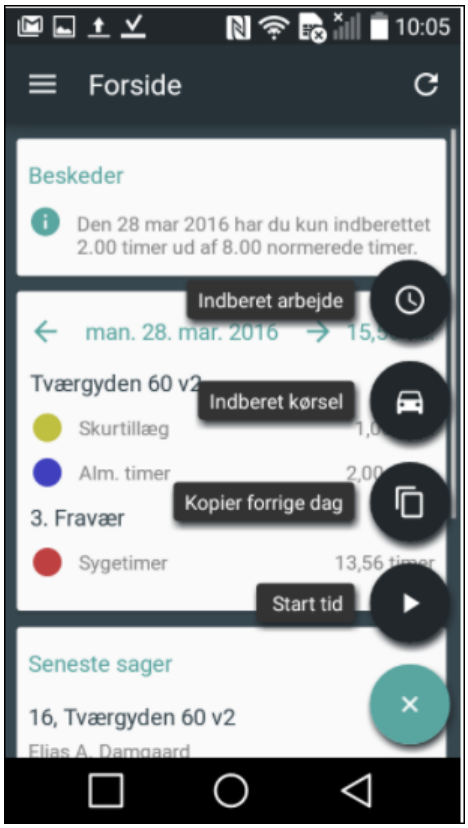


Рис. 1.7. Начальный экран

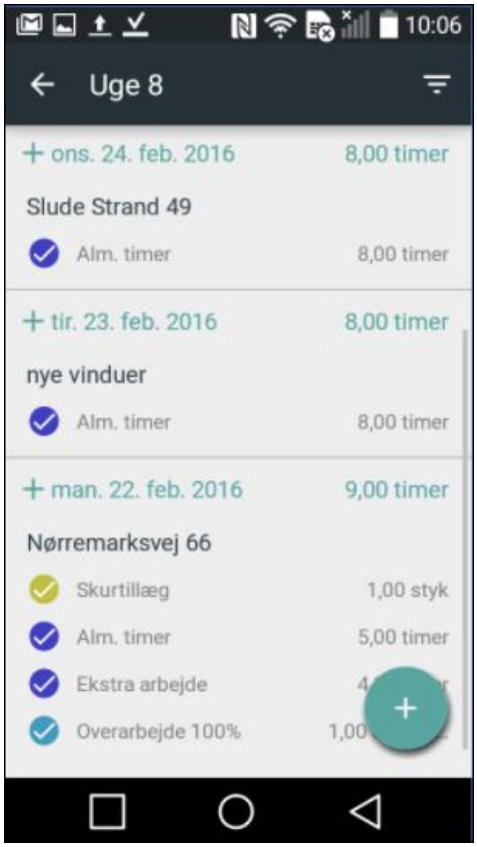


Рис. 1.8. Включение таймера

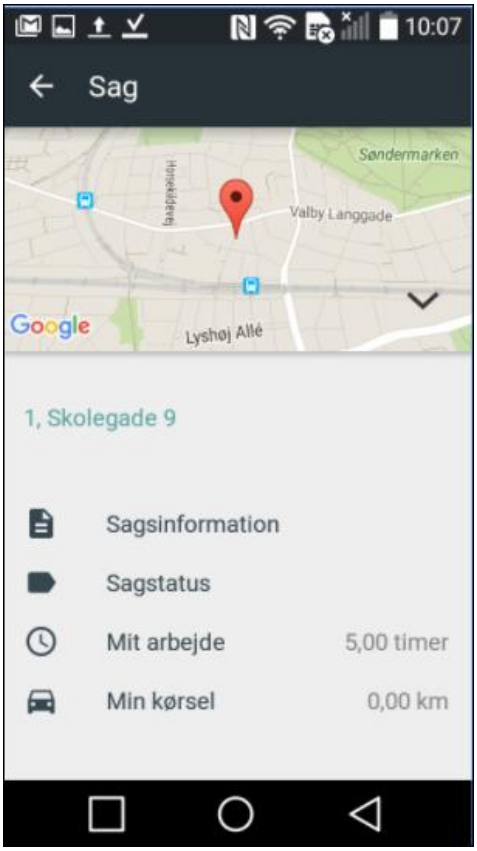


Рис. 1.9. Навигация

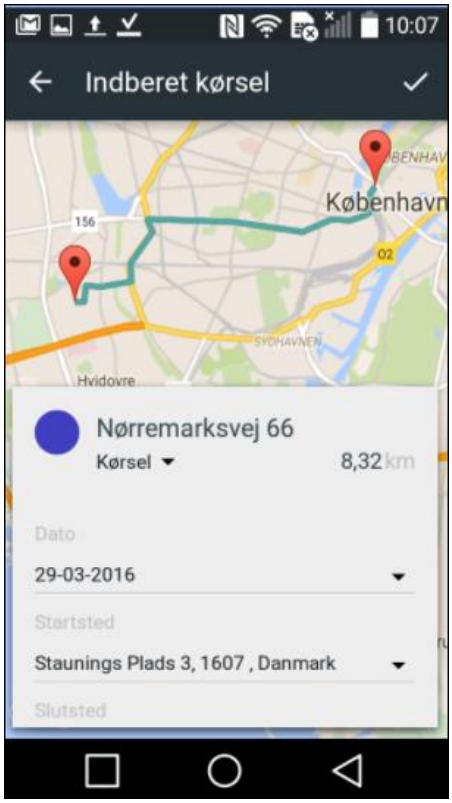


Рис. 1.10. Построение маршрута

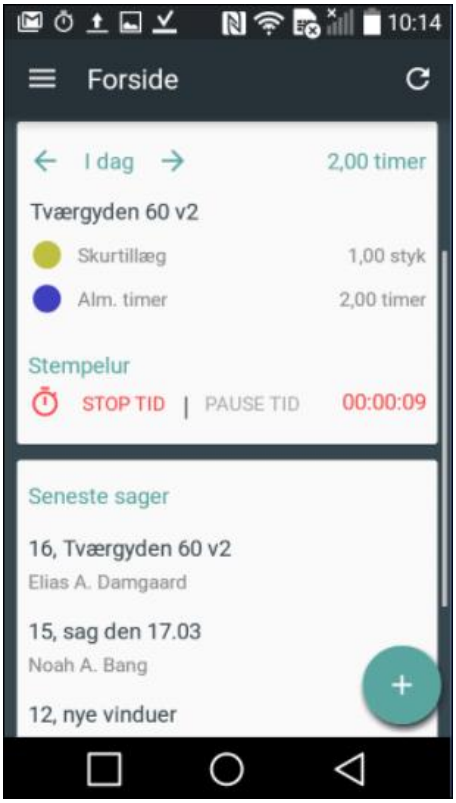


Рис. 1.11. Время работы

Описание приложений заимствовано с сайта Play Маркет.

Существует несколько популярных мобильных приложений, похожих по наполнению. Можно сказать, что каждый из них не имеет никакой четкой классификации. Любое из них является уникальным с оригинальным дизайном и контентом, что позволяет почерпнуть новые идеи для разработки мобильного приложения.

Невозможно отрицать, что в сфере информационных технологий вчерашние инновации – это сегодняшние привычные вещи. Чтобы идти «в ногу со временем», специалистам в этой сфере необходимо использовать концепцию постоянного обучения, и ключом к этому является своевременное освоение наиболее значительных, распространенных и перспективных технологий. Технологии разработки под операционную систему Android для планшетов и смартфонов относятся именно к таковым. На основании приведенных сведений о международных удостоверениях ISIC и современном рынке мобильных приложений можно сделать вывод о целесообразности разработки мобильного приложения на базе операционной системы Android.

ГЛАВА II Разработка приложения под платформу ANDROID

2.1 Устройство платформы ANDROID

Платформа Android объединяет операционную систему, построенную на основе ядра ОС Linux, промежуточное программное обеспечение и встроенные мобильные приложения. Разработка и развитие мобильной платформы Android выполняется в рамках проекта AOSP (Android Open Source Project) под управлением OHA (Open Handset Alliance).

Архитектура системы Android (рисунок 2) представляет собой полный программный стек, в котором можно выделить следующие уровни:

Уровень приложений (Applications) - набор предустановленных базовых приложений, клиенты электронной почты и SMS, календарь, различные карты, браузер, программа для управления контактами и многое другое. Все приложения, запускаемые на платформе Android, написаны на языке Java.

Уровень каркаса приложений (Application Framework) Android позволяет использовать всю мощь API, используемого в приложениях ядра. Архитектура построена таким образом, что любое приложение может использовать уже реализованные возможности другого приложения при условии, что последнее откроет доступ на использование своей функциональности. Таким образом, архитектура реализует принцип многократного использования компонентов ОС и приложений.

Основой всех приложений является набор систем и служб:

1. Система представлений (View System) – это богатый набор представлений с расширяемой функциональностью, который служит для построения внешнего вида приложений, включающий такие компоненты, как списки, таблицы, поля ввода, кнопки и т.п.

2. Контент-провайдеры (Content Providers) – это службы, которые позволяют приложениям получать доступ к данным других приложений, а также предоставлять доступ к своим данным.

3. Менеджер ресурсов (Resource Manager) предназначен для доступа к строковым, графическим и другим типам ресурсов.

4. Менеджер извещений (Notification Manager) позволяет любому приложению отображать пользовательские уведомления в строке статуса.

5. Менеджер действий (Activity Manager) управляет жизненным циклом приложений и предоставляет систему навигации по истории работы с действиями.

Набор библиотек и среда исполнения (Libraries & Android Runtime). Платформа Android включает набор C/C++ библиотек, используемых различными компонентами ОС. Для разработчиков доступ к функциям этих библиотек реализован через использование Application Framework. Ниже представлены некоторые из них:

1. System C library — BSD-реализация стандартной системной библиотеки C (libc) для встраиваемых устройств, основанных на Linux.

2. Media Libraries – библиотеки, основанные на PacketVideo‘s OpenCORE, предназначенные для поддержки проигрывания и записи популярных аудио- и видео- форматов (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG и т.п.).

3. Surface Manager – менеджер поверхностей управляет доступом к подсистеме отображения 2D- и 3D- графических слоев.

4. LibWebCore – современный движок web-браузера, который предоставляет всю мощь встроенного Android-браузера.

5. SGL – движок для работы с 2D-графикой.

6. 3D libraries – движок для работы с 3D-графикой, основанный на OpenGL ES 1.0 API.

7. FreeType – библиотека, предназначенная для работы со шрифтами.

8. SQLite – мощный легковесный движок для работы с реляционными БД.

В состав Android входит набор библиотек ядра, которые предоставляют большую часть функциональности библиотек ядра языка Java.

Android Runtime - платформа использует оптимизированную, регистр-ориентированную виртуальную машину Dalvik, в отличии от нее стандартная виртуальная машина Java – стек-ориентированная. Каждое приложение запускается в своем собственном процессе, со своим собственным экземпляром виртуальной машины. Dalvik использует формат Dalvik Executable (*.dex), оптимизированный для минимального использования памяти приложением. Это обеспечивается такими базовыми функциями ядра Linux, как организация поточной обработки и низкоуровневое управление памятью

Базовый уровень (Linux Kernel). Android основан на ОС Linux, тем самым платформе доступны системные службы ядра, такие как управление памятью и процессами, обеспечение безопасности, работа с сетью и драйверами. Также ядро служит слоем абстракции между аппаратным и программным обеспечением.

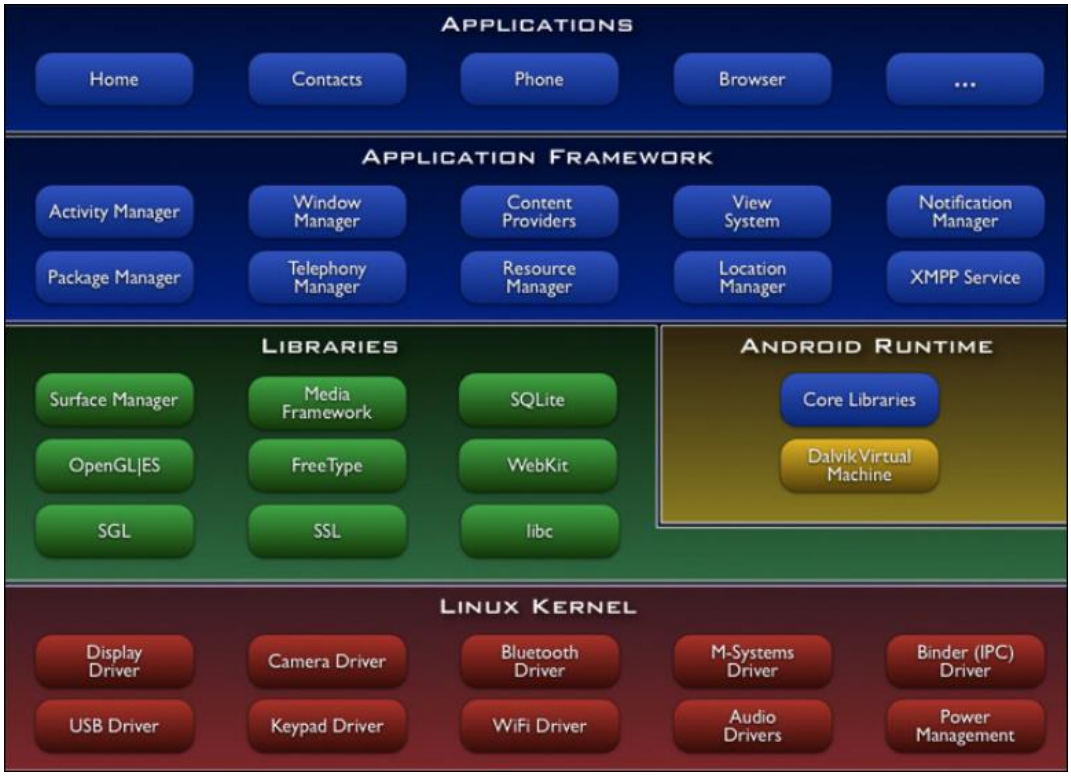


Рис. 2 –Архитектура системы Android.

Приложения для Android в своей работе использует окна (аналогично Windows), однако в данной системе вышеуказанные окна носят иное название - Activity. Как и в Windows, каждое окно имеет свой жизненный цикл и свои особенности. При создании нового окна вызывается метод onCreate(), при разработке данный метод переназначается далее в нем происходит инициализация приложения и его компонентов. Затем вызываются методы onStart() и onResume(). Эти методы вызываются перед отображением окна при его создании, либо восстановлении (при переключении из другого приложения, при разворачивании свернутого приложения). При сворачивании вызываются методы onPause() и onStop().

При закрытии приложения и окна вызывается onDestory(), в данном методе можно сохранить пользовательские данные и параметры. Полное описание и очередность вызова методов можно найти на официальном сайте.[18] Общая схема жизненного цикла приложения для Android представлена на рисунке 3.

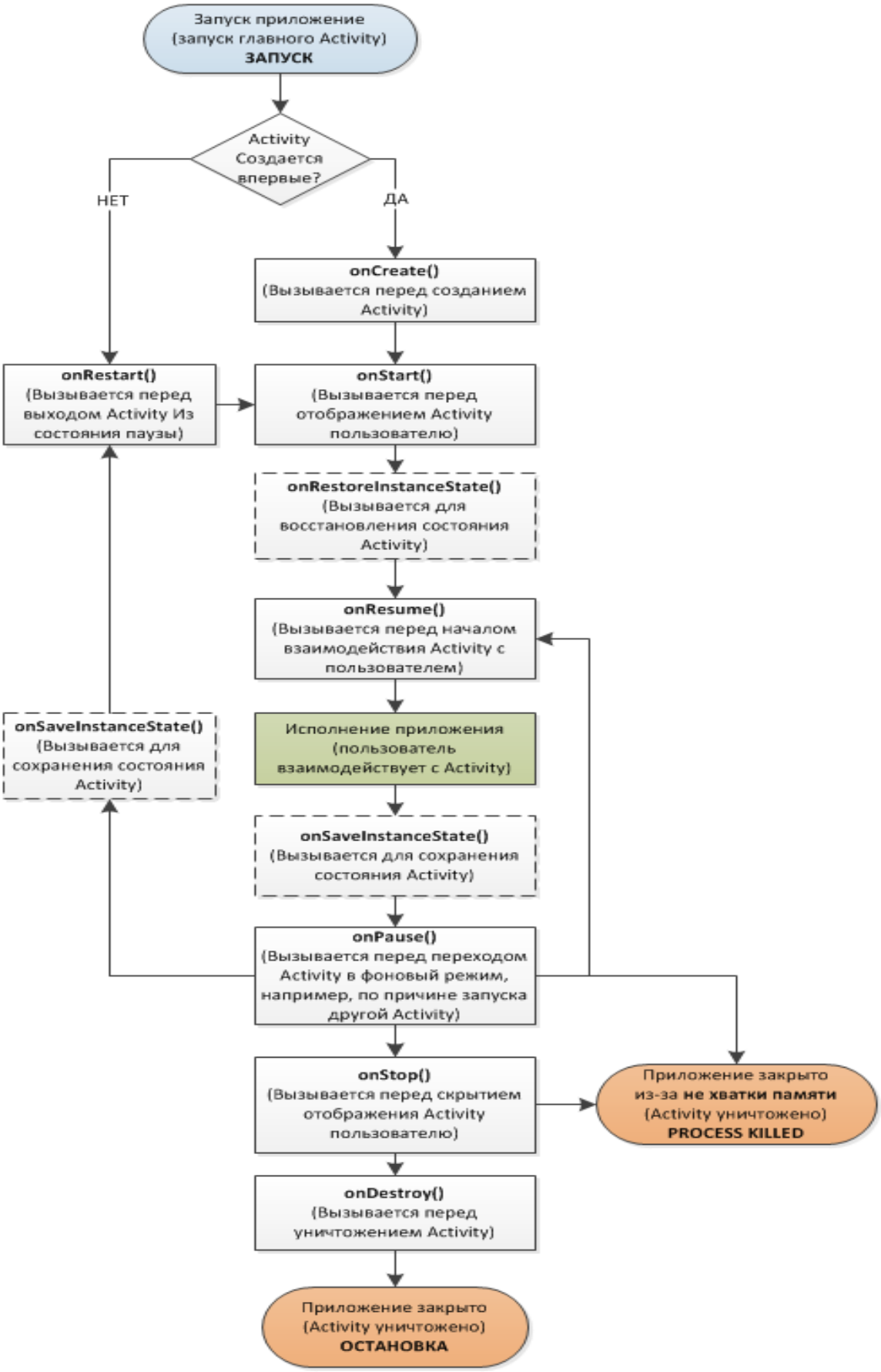


Рис. 3. Жизненый цикл Android

Корневой каталог каждого приложения под Android должен содержать файл AndroidManifest.xml (в точности с таким названием). Манифест приложения содержит всю необходимую информацию, используемую системой для запуска и выполнения приложения. Основная информация, содержащаяся в манифесте:

- имя Java пакета приложения, которое используется как уникальный идентификатор приложения;
- описание компонентов приложения: активностей, сервисов, приемников широковещательных сообщений и контент-провайдеров, которые составляют приложение. Для каждого компонента приложения определено имя соответствующего класса и объявлены их основные свойства (например, с какими сообщениями-намерениями они могут работать). Эта информация позволяет системе Android узнать какие компоненты и при каких условиях могут быть запущены;
- определение процессов, в которых будут выполняться компоненты приложения;
- объявление полномочий, которыми должно обладать приложение для доступа к защищенным частям API и взаимодействия с другими приложениями;
- объявление полномочий, которыми должны обладать другие приложения для взаимодействия с компонентами данного;
- список вспомогательных классов, которые предоставляют информацию о ходе выполнения приложения. Эти объявления содержатся в манифесте пока идет разработка и отладка приложения, перед публикацией приложения они удаляются;
- определение минимального уровня Android API для приложения;
- список библиотек связанных с приложением.

В файле манифеста только два элемента: «manifest» и «application» являются обязательными и при этом встречаются ровно по одному разу. Остальные

элементы могут встречаться несколько раз или не появляться совсем, в этом случае манифест определяет пустое приложение.

2.2 Разработка мобильного приложения для учета времени

Средой для разработки мобильного приложения была выбрана Android Studio — это официальная среда разработки под систему Android. Существует множество сред разработки, и эта среда была выбрана за ее доступность и рекомендациям среди программистов, благодаря ее удобному графическому интерфейсу и средств оптимизации.

Android Studio - интегрированная среда разработки приложений, созданная компанией Google для операционной системы Android. Предоставленный продукт призван обеспечить разработчиков новыми инструментами для создания приложений. При создании нового проекта в Android Studio, изображена структура проекта со всеми нужными файлами, содержащимися в каталоге SDK. Этот переход к системе управления Gradle сообщает процессу разработки еще большую гибкость, Android Studio разрешает нам увидеть возможные визуальные изменения, которые совершает в реальном времени в приложении. Также можно увидеть, как приложение будет в одно то же время смотреться на разных устройствах под управлением Android, с разнообразными настройками и разрешением экрана. Studio обладает новыми инструментами для создания и маркировки кода. Что позволяет нам не потеряться в проекте, когда имеем дело с большим количеством кода. В программе также задействована функция перетаскивания, с помощью которой можно перемещать компоненты средствами пользовательского интерфейса. Ко всему прочему среда разработки имеет в своем распоряжении функцию Google Cloud Messaging. Эта функция позволяет нам отсылать данные с сервера на Android-

устройства через облако. Это прекрасный способ отправлять push-уведомления приложениям. Существует возможность с помощью программы локализовать приложения. Что позволяет нам писать код, и контролировать наше приложение.

Возможности Android Studio:

- Надежная и простая среда разработки.
- Есть возможность свободно проверить продуктивность приложения на различных типах устройств.
- Помощники и шаблоны для многих элементов программирования для Android.
- Многофункциональный редактор с массой дополнительных инструментов, способствующих увеличению разработки приложений.

Установка Android Studio.

Android Studio необходимо скачать.[20] В комплекте компонентов: будет все необходимое — сама IDE, Android Emulator, Android SDK, инструменты для проведения тестов и отладки приложений. Недостающие средства в комплекте, инсталлятор поможет скачать самостоятельно. Чтобы создать приложение, нам нужно в Android Studio создать проект. При создании проекта, в нем создается модуль (см. рисунок 4). Кроме выбора типа устройств, надо выбрать минимальную версию системы (Minimum SDK), под которую будет работать приложение. На данный момент Гугл поддерживает версии, начиная с API 7, выпуская специальные библиотеки совместимости для старых устройств. Если щёлкнуть по ссылке «Help me choose,» то откроется окно с графиком [Приложение 1]. В модуле пишем код, прописываем визуально экраны нашего приложения. При запуске модуля мы получаем «сырое» приложение. Модуль по сути и есть наше приложение. А проект - контейнер для модуля.

Рис. 4. Проект-модуль.

При создании нового проекта (рисунок 5) в программе, полностью доступна вся структура программы вместе со всеми файлами, что несомненно дает возможность максимально эффективно и продуманно организовать сам процесс разработки. Очень удобно внедрена возможность показывать вносимые изменения и дополнения, выводятся визуально в режиме настоящего времени, происходят преобразования в точной зависимости от заданных алгоритмов.

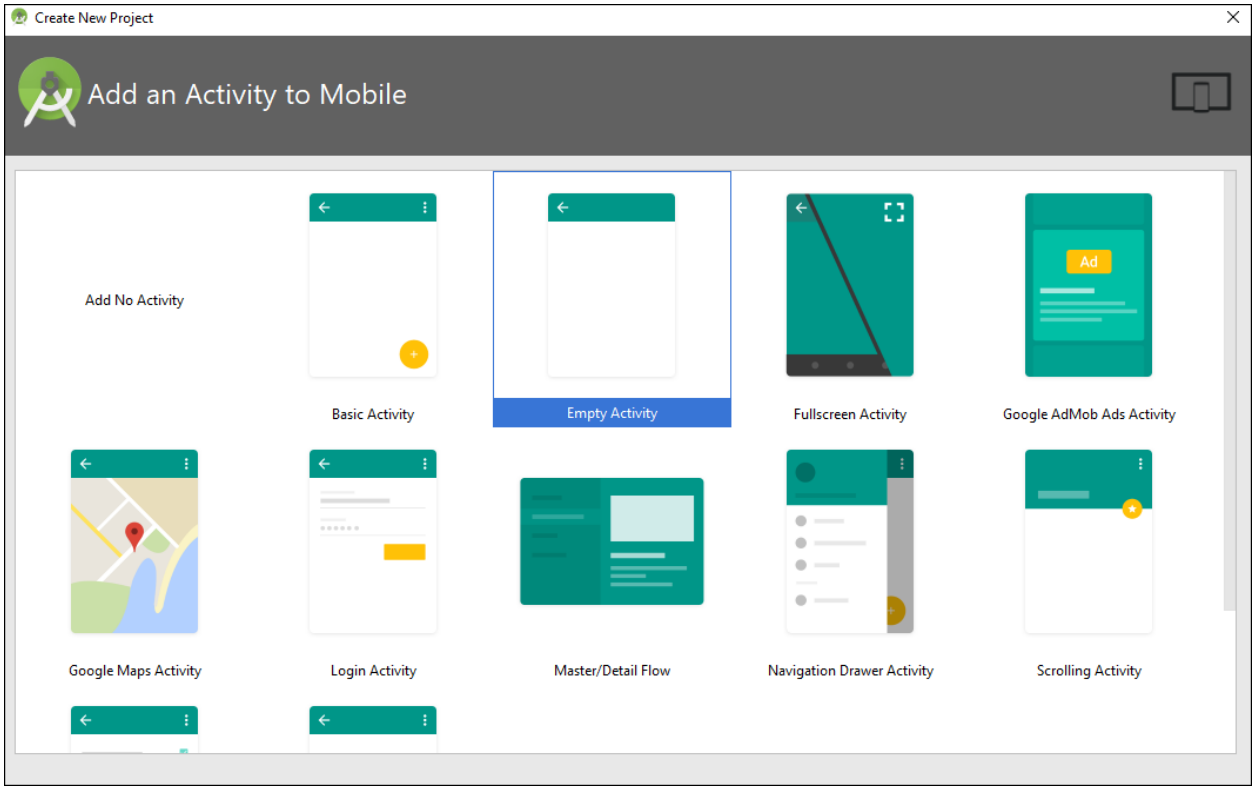


Рис. 4. Создание нового проекта в Android Studio

Важным фактом является то, что Android Studio предоставляет возможность заниматься разработкой программ практически для всех версий ОС Android, начиная с релиза, и заканчивая последней на текущий момент версией 8.0. Существует инструмент для предварительной оценки внешнего вида программы для различных устройств, например, смартфон или планшет. Удобная подсветка кода, позволяет легко ориентироваться в больших объемах кода. Кроме того, большим плюсом является изменение отдельных компонентов методом простого перетаскивания в то место, где пользователь захочет его видеть, данная функция существенно облегчает процесс программирования. Android Studio обладает инструментом Google Cloud Messaging, главной задачей которого является настройка отправлений всплывающих уведомлений для приложений, задействуя при этом облачные сервисы под управлением ОС Android. В итоге, можно сделать вывод о многозадачности Android Studio. Одним из преимуществ данной среды является массивное подробное тестирование приложения перед его выпуском. Тестирование провидится во всевозможных режимах работы, что позволяет исправить множественные недочеты.

Свойства программы Android Studio:

- Функции производительности
- Быстрое исправления ошибок рендеринга..
- Возможность построения проектов на базе Gradle.
- Наличие всплывающих подсказок.
- Идентификация областей кода.
- Работа в редакторе слоев, в котором есть возможность управлять всеми элементами методом перетаскивания мышью.
- Мастер шаблонов, быстро создать стандартные дизайны.
- Наличие инструментов, проверяющих возможности совместимости различных приложений.

Во время создания нового проекта в Android Studio в проекте заводится основная структура проекта с исходным кодом, папками и файлами по стандарту. Android Studio предоставляет ряд шаблонов для разных обстоятельств, но самыми частыми являются Basic Activity и Empty Activity. Это самые подходящие шаблоны для создания подавляющего большинства приложений. Empty Activity выражается просто пустым экраном, а Basic Activity имеет ряд минимальных элементов, для облегчения формирования новых проектов. Шаблоны проектов предоставлены на рисунке 6.

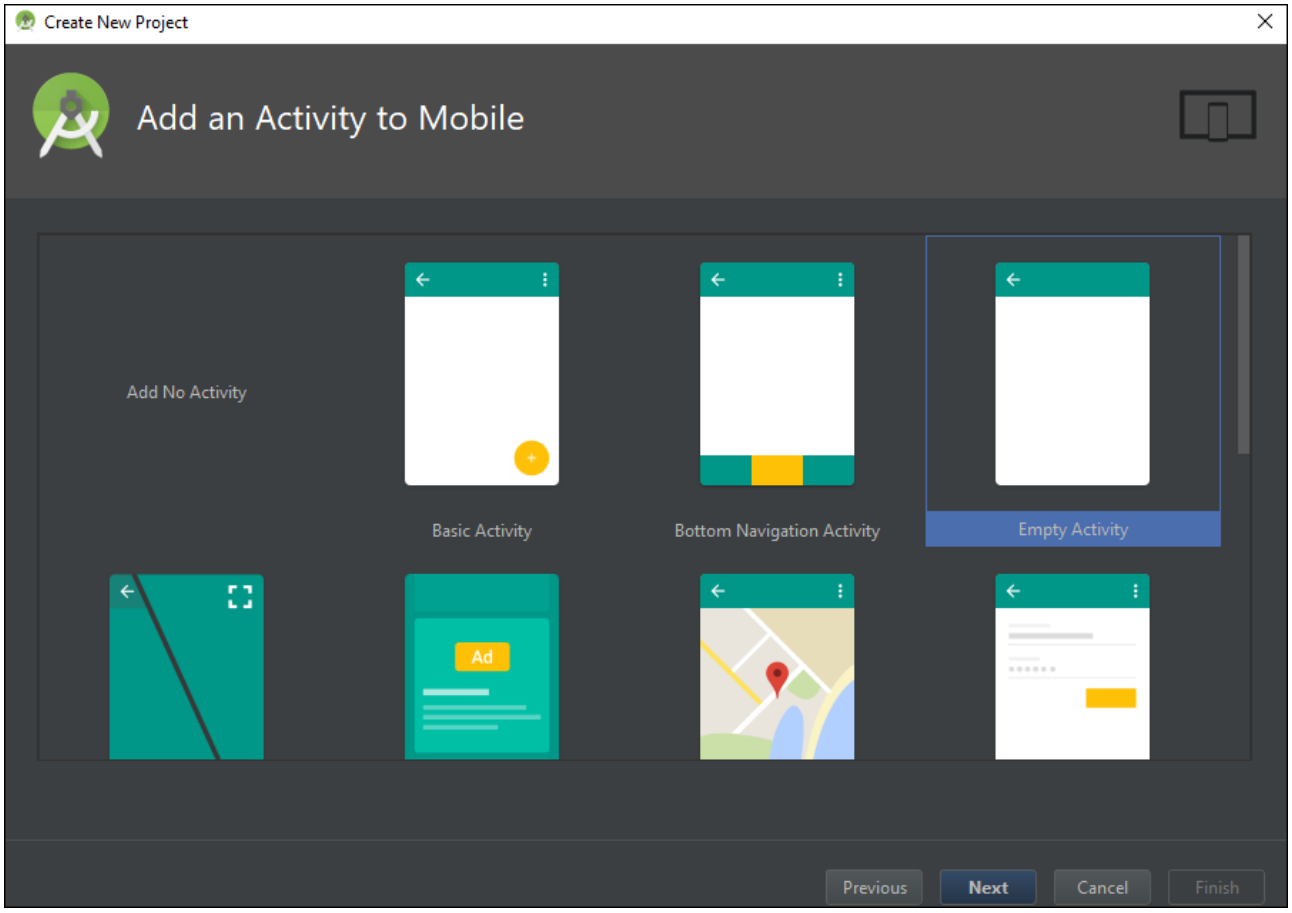


Рис. 6. Шаблоны проектов.

В папке manifests находится файл AndroidManifest.xml (рисунок 7) в котором можно изменять название приложения, его значок, стартовую страницу, список Activity и другие технические настройки.

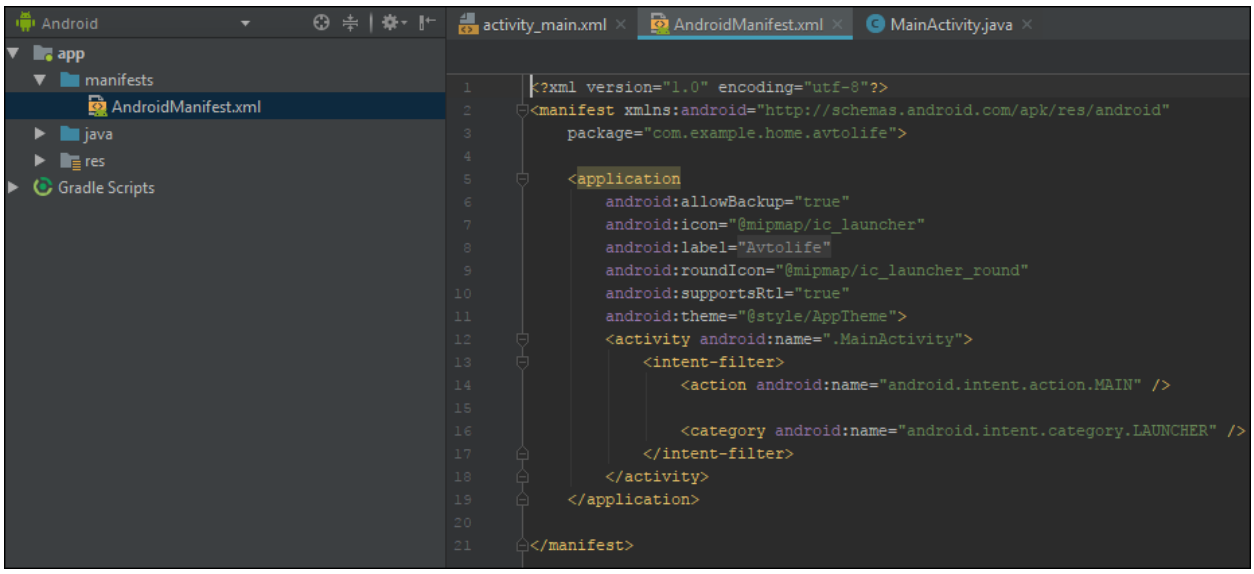


Рис. 7. Расположение файла AndroidManifest.xml

Код прописывается на языке java, интерфейсы и классы находятся в папке java. В папке находятся другие подпапки и файлы, из которых для нас важнейшими являются каталоги androidTest и test предназначенные для хранения файлов тестов приложения. А собственно исходные коды располагаются в папке главном файле с названием проекта.

Название обусловлено тем, что в название проекта автоматически добавляется имя пользователя, создавшего проект. Стандартно в этом каталоге находится файл класса MainActivity, который запускается по умолчанию при старте приложения (см. рисунок 8).

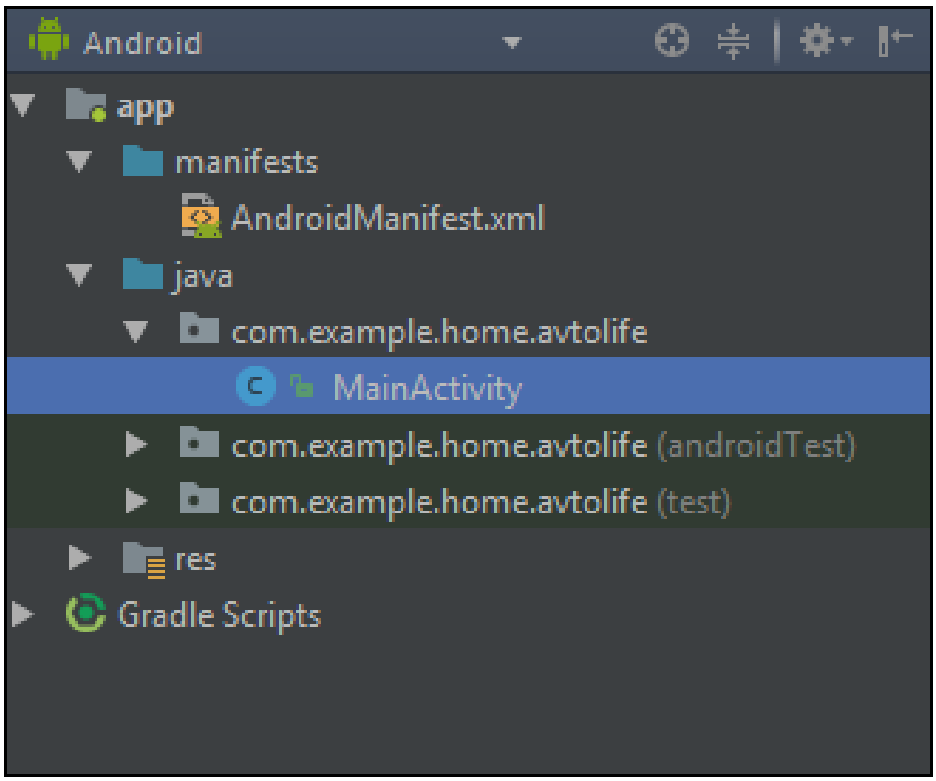


Рис. 8. Папки классов

В папке «res» (см. рисунок 9) лежат все ресурсы, которые будет использовать приложение. Они распределены по подпапкам:

• drawable – для графических файлов растровых и векторных форма-тов;

• layout – для файлов разметки Activity формата xml. По умолчанию здесь есть два файл activity_main.xml, который определяет интерфейс для стандартной activity – MainActivity;

• menu для разметки различных меню и подменю;

• mipmap – По сути, это замена ресурсам drawable для значков приложения, появилось в более поздних версиях android;

• values – содержит текстовые переменные на различных языках и файлы настроек;

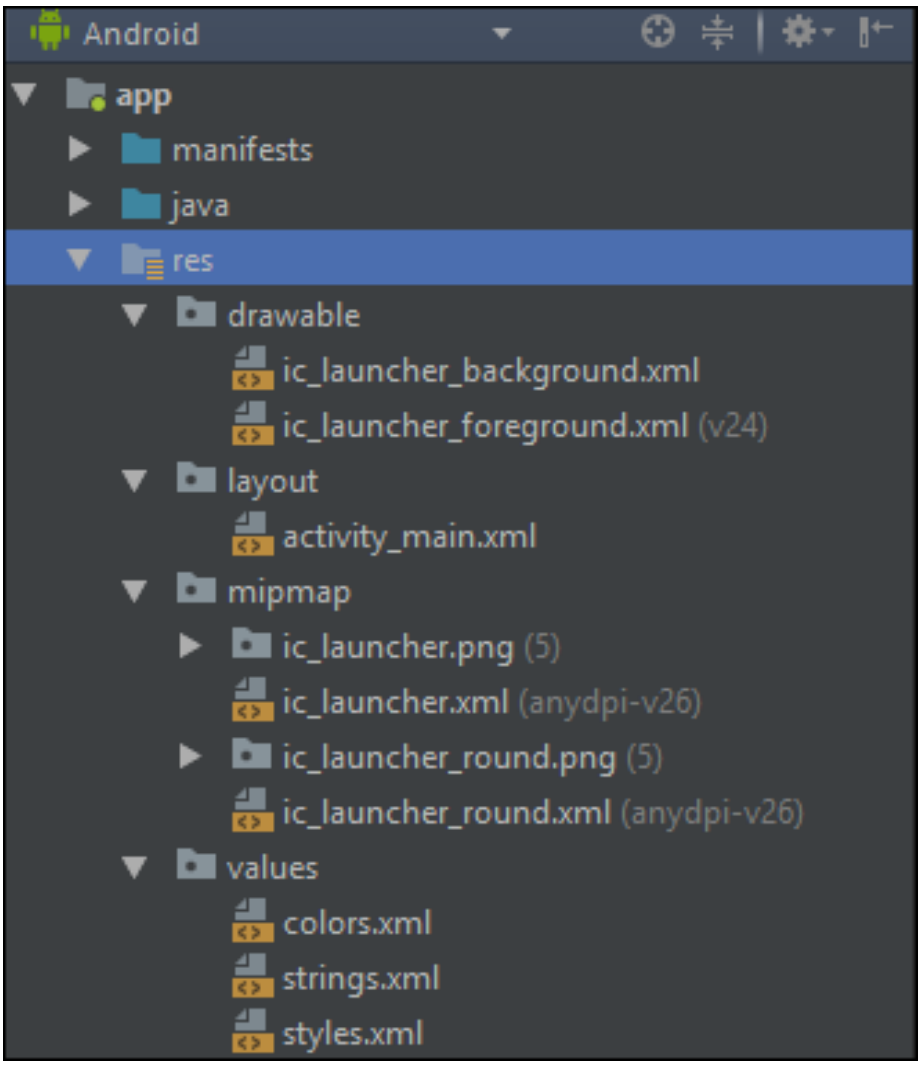


Рис. 9. Папка «res»

Файл Gradle Scripts (рисунок 10) содержит информацию, которая используется при построении проекта. Здесь можно подключить дополнительные модули и расширения к проекту.

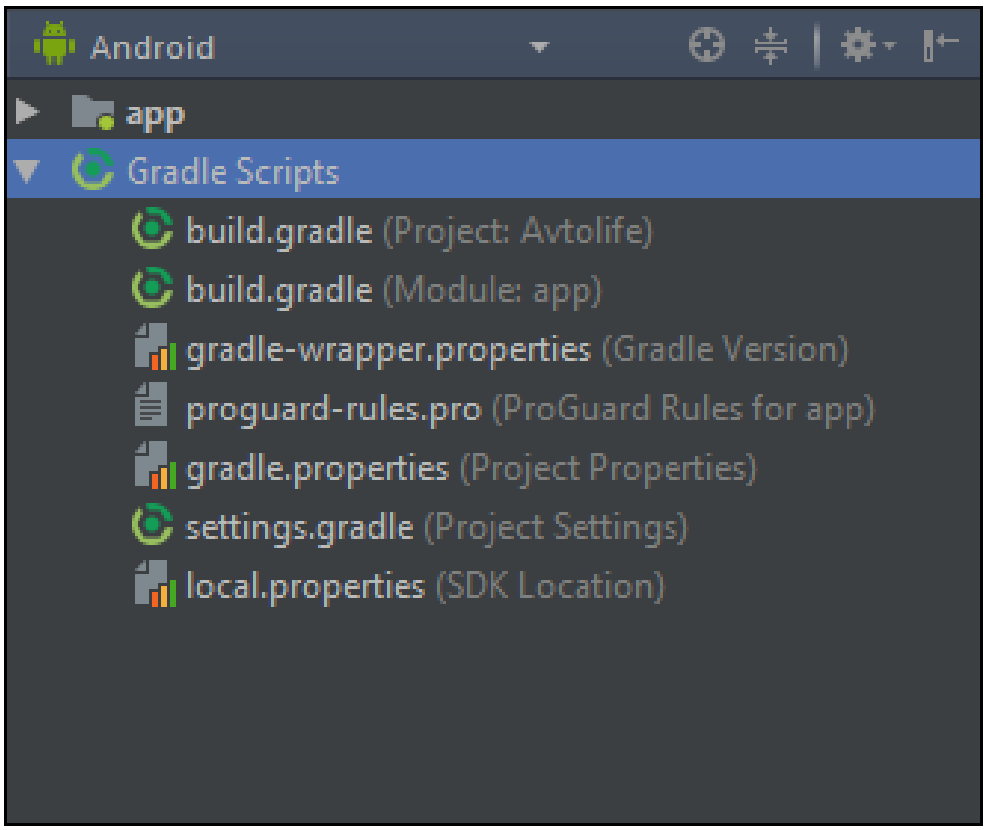


Рис. 10. Папка Gradle Scripts

Приложение было разработано исходя из схемы расположения его частей. Приложение «AvtoLife» состоит из четырех экранов. Такая структура наиболее удобна во время использования приложения. И имеет хорошие результаты производительности и навигации.

Начальный экран приложения имеет три области.

Первая область разделена на две части:

- выбор выпадающего меню (шторка)
- логотип приложения

Вторая область позволяет пользователю зарегистрироваться в приложении.

- введите логин
- введите пароль

Третья область «кнопка», при нажатии которой пользователь получает доступ в систему.

Схематичное расположение областей указано на рисунке 11. Фрагмент кода [Приложение 2]

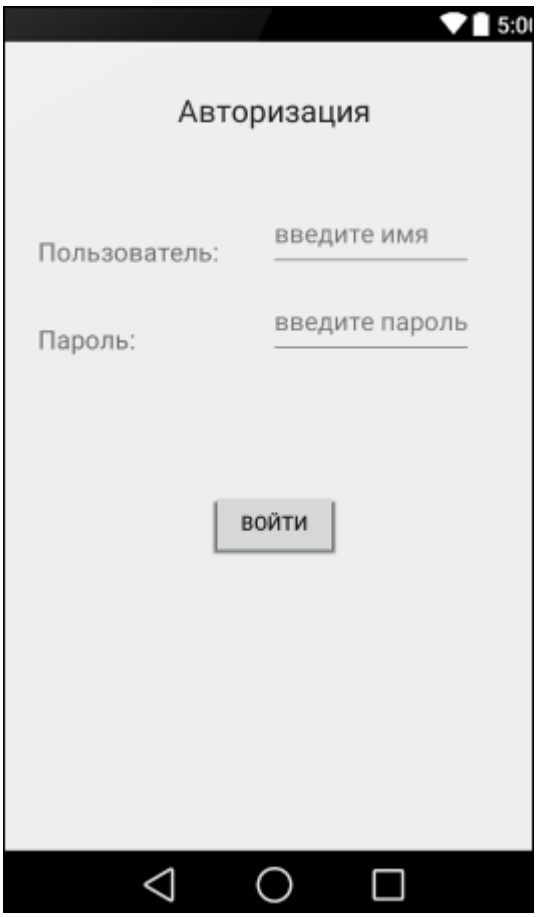


Рис. 11. Начальный экран

Второй экран разделен на две области. Первая область так же разделана на две части:

- выбор выпадающего меню
- логотип приложения

Во второй области расположен календарь, при выборе даты пользователь может добавить новый рабочий элемент.

Под календарем находиться подсказка, что нужно выбрать дату.

Схематичное расположение областей указано на рисунке 12

Рис. 12. Второй экран

Третий экран. В верхней области находиться

- выбор выпадающего меню

- Табель времени

Центральная область:

Кнопка «старт»- начинает отчет рабочего времени.

Кнопка «обед» - начинает отчет когда пользователь уходит на обед, по умолчанию время 1 час.

Кнопка «перерыв» - начинает отчет во время перерыва, по умолчанию время 15 минут.

Кнопка «конец дня» - при нажатии пользователя, фиксируется время окончания рабочего дня.

Кнопка «рабочее время» - высчитывается автоматически. Рисунок 13

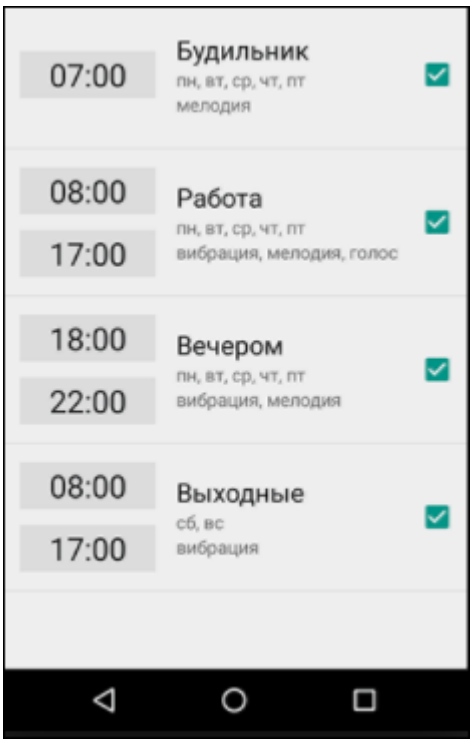


Рис. 13. Третий экран

Четвертый экран – Рабочий день

«Проект» - пользователь выбирает направление работы из выпадающего списка:

- Норма
- Больничный
- Отпуск
- Ночной
- Выходной
- Командировка

«График» - пользователь выбирает по какому графику ему работать. В меню выпадающего списка, можно выбрать наиболее подходящий график :

- Дневной
- Месячный
- Сменный

- Недельный

«Начало рабочего дня» - отмечается время начала смены, утро, день или ночь.

«Место отправления» - пользователь заполняет самостоятельно, адрес пункта отправления.

«Место назначения» - пользователь заполняет самостоятельно, адрес куда должен прибыть пользователь.

«Добавить промежуточное место» - позволяет пользователю добавить места в которых он останавливался между пунктами отправления и назначения.

Выпадающее меню (шторка), три горизонтальные полоски. В официальной документации Гугла называется «гамбургером»

Верхняя область:

- Рабочий календарь
- Редактировать рабочий день
- Удалить рабочий день.
- Список проектов

Нижняя область позволяет работать с данными

- Сохранить в файл
- Отправить на почту
- Импортировать на карту памяти
- Экспортировать

Android это уникальная операционная система. Разработчик приложений должен знать ее особенности и нюансы для получения хорошего результата. Сами приложения могут быть разработанными для всех платформ, то есть кросс программы, и персонально для каждой системы. Как показывает практика, возможности индивидуального ПО гораздо шире, что делает целесообразнее устанавливать соответствующее обеспечение, а следовательно и разрабатывать необходимые приложения.

Получен глубокий опыт и понимание принципов стороннего взаимодействия систем в операционной системе Android. Работая над проектам, я понимаю важность своей работы, так как это несет в себе большую значимость для меня в IT сфере.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы было создано мобильное приложение, позволяющее пользователю вести учет своего рабочего времени. В приложении есть возможность выбрать сменный график, если он не нормирован. Так же приложение позволяет вести учет времени, если норма часов превышает стандарты. Вся переработка учитывается.

Была выбрана среда разработки Anroid Studio, как самая популярная среди разработчиков. Студио находиться в свободном доступе, и является официальным средством разработки Андроид приложений. Так же была выбрана операционная система мобильного устройства Андроид для работы приложения.

В ходе выполнения выпускной работы были решены следующие задачи:

проанализированы предметная область и аналогичные проекты в магазине приложений «Google Play»;

подготовлен проект приложения;

написано мобильное приложение под управлением операционной системы Android;

Изучена разработка приложений под Android на Java;

Изучены основы язык Java для программирования приложений.

Результатом работы является создание мобильного приложения для учета рабочего времени

Разработанный проект удовлетворяет всем требованиям технического задания.

Следует считать, что задачи выпускной квалификационной работы полностью решены и цель исследования достигнута.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

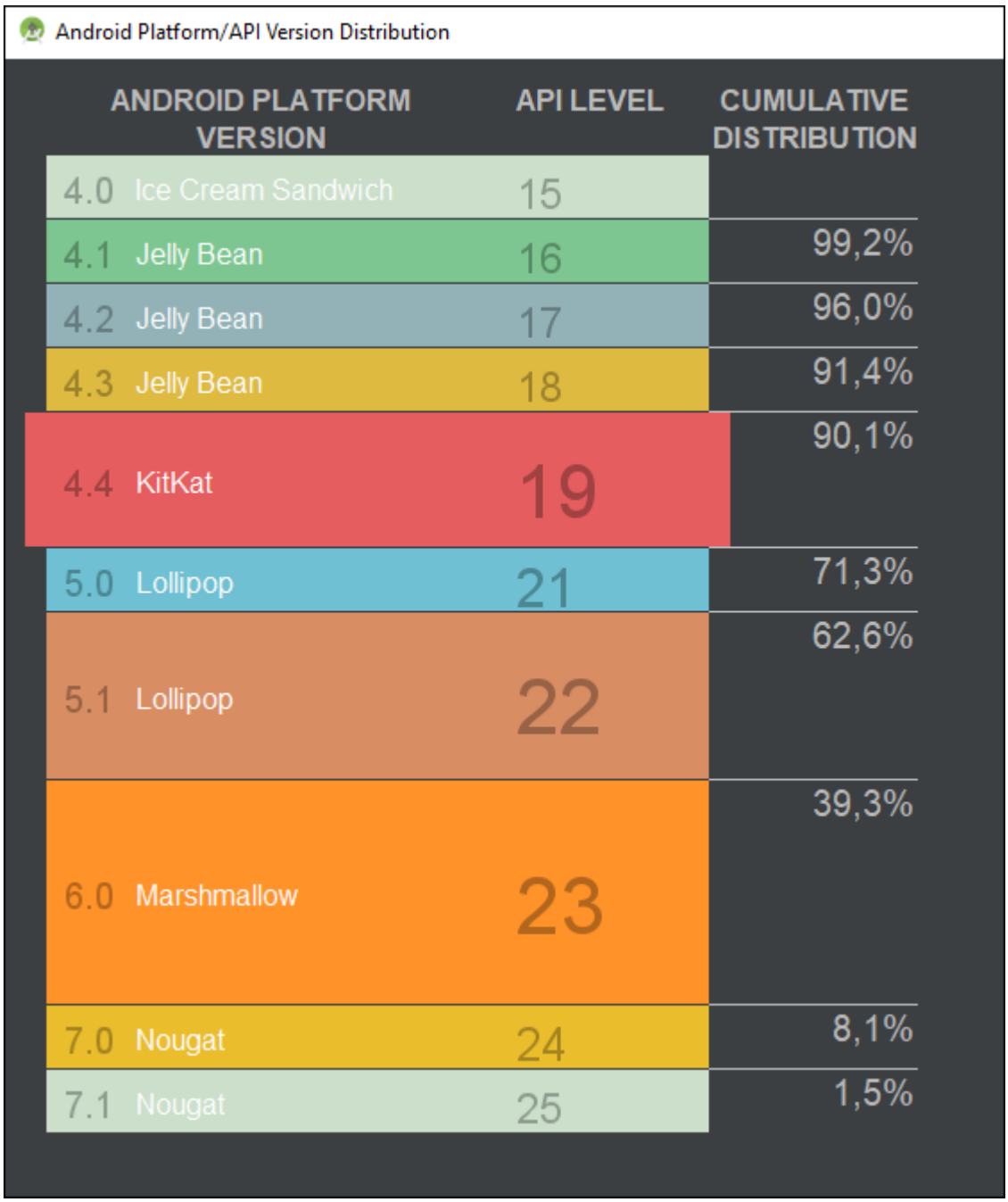
1. Android для разработчиков. 3-е изд. — СПб.: Питер, 2016. — 512 с.: ил. — (Серия «Библиотека программиста»).
2. Java 8: руководство для начинающих, 6-е изд. : Пер. с англ. - М.ООО "И.Д. Вильяме", 2015. - 720 с.: ил. - Парал. тит. англ.
3. Блэйк Мик /Программирование под Android . - СПб.: Санкт-Петербург, 2012. - 496 с.
4. Брюс Эккель «Философия Java» (4-е полное издание): СПБ.: Питер,2015 – 1168 с.
5. Дорнин Л. Google Android: программирование для мобильных устройств. — СПб.: БХВ-Петербург, 2012. — 448 с.: ил.+CD-ROM — (Профессиональное программирование)
6. Дэрси Л. Разработка приложений для Android-устройств. Базовые принципы /Л. Дэрси, Ш. Кондер – Том 1. – Москва: Эксмо, 2014. – 598 с.
7. Жвалевскийм А. Смартфоны Android без напряга. Руководство пользователя / . - СПб.: Санкт-Петербург, 2012. - 224 с.
8. Колисниченко Д. Программирование для Android. Самоучитель /. - СПб.: Санкт-Петербург, 2013. - 736 с.
9. Медникс З.. Программирование под Android. Издательство Питер, 2012. – 496
10. Программирование для Android. Самоучитель / Денис Колиснеченко . - СПб.: Санкт-Петербург, 2013. - 272 с.
11. Рето Маейр /Android 2: программирование приложений для планшетных компьютеров и смартфонов : [пер. с англ.] /. — М. : эскмо, 2013. — 672 с. — (Мировой компьютерный бестселлер).
12. Харди Б., Филлипс Б. Программирование под Android.– Спб: Питер, 2013. – 592с.

13. Бурнет Э. Привет, Android! Разработка мобильных приложений СПб.: Питер, 2012.
14. Android [Электронный ресурс]. – Режим доступа: http://www.android.com/
15. Android Studio [Электронный ресурс] - https://developer.android.com/studio/index.html
16. Habrahabr.ru [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/posts/programming/
17. Java — Учебник для начинающих программистов [Электронный ресурс] http://proglang.su/java
18. Oracle [Электронный ресурс]. - https://www.oracle.com/index.html
19. Start Android – учебник по Android для начинающих и продвинутых [Электронный ресурс]. – Режим доступа: http://startandroid.ru/ru/
20. Блог разработчиков Android [Электронный ресурс] - https://android-developers.googleblog.com/2014/07/learn-to-think-like-android-developer.html
21. Книги по Java [Электронный ресурс] - https://lyapidov.ru/category/programming/java/
22. Компоненты приложения [Электронный ресурс] - https://developer.android.com/guide/components/activities/index.html#Lifecycle
23. Мобильное приложение, что такое, определение, новости, статьи, видео [Электронный ресурс]. – https://indicator.ru/tags/ mobilnoe-prilozhenie/
24. Основные этапы разработки мобильных приложений [Электронный ресурс] – Режим доступа: https://spark.ru/startup/componentix/blog/4499/ osnovnie-etapi-razrabotki-mobilnih-prilozhenij
25. Официальная документация по Android [Электронный ресурс]. - https://developer.android.com/guide/

26. Приложение Google I / O 2017 для Android [Электронный ресурс] - https://github.com/google/iosched
27. Программирование для android, java [Электронный ресурс] http://davidmd.ru/%D1%83%D1%80%D0%BE%D0%BA%D0%B8-%D0%BF%D0%BE-android/
28. Программирование на Java [Электронный ресурс] - http://study-java.ru/category/uroki-java/
29. Руководство разработчика [Электронный ресурс] - http://mybiblioteka.su/tom3/7-59852.html
30. Русская документация Android [Электронный ресурс] - http://easyandroid.ru/index.php?p=721
31. Тестирование android приложений с помощью реальных устройств [Электронный ресурс] - http://www.fandroid.info/testirovanie-android-prilozhenij-s-pomoshhyu-realnyh-ustrojstv/
32. Уроки по Java [Электронный ресурс] - https://javarush.ru/
33. Уроки программирования [Электронный ресурс] - http://learn-android.ru/index.html
34. Уроки программирования от Google [Электронный ресурс] - https://developer.android.com/training/index.html

ПРИЛОЖЕНИЕ 1

ПРОЦЕНТ ИСПОЛЬЗОВАНИЯ ВЕРСИЙ



ПРИЛОЖЕНИЕ 2

Фрагмент кода

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">

<TextView
    android:id="@+id/Login"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="20dp"
    android:text="Авторизация"
    android:textAppearance="?android:attr/textAppearanceLarge" />

<TextView
    android:id="@+id/user_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/Login"
    android:layout_marginTop="75dp"
    android:layout_marginLeft="10dp"
    android:text="Пользователь:"
    android:textAppearance="?android:attr/textAppearanceMedium" />

<EditText
    android:id="@+id/edit_user"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@+id/user_text"
    android:layout_marginLeft="35dp"
    android:layout_toRightOf="@+id/user_text"
    android:hint="введите имя " >
    <requestFocus />
</EditText>

<TextView
    android:id="@+id/password_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/user_text"
    android:layout_below="@+id/user_text"
    android:layout_marginTop="40dp"
    android:text="Пароль:"
    android:textAppearance="?android:attr/textAppearanceMedium" />

<EditText
    android:id="@+id/edit_password"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@+id/password_text"
    android:layout_alignLeft="@+id/edit_user"
    android:layout_alignRight="@+id/edit_user"
    android:hint="введите пароль"
    android:inputType="textPassword" />

<TextView
    android:id="@+id/attempts"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/password_text"
    android:layout_below="@+id/password_text"
    android:layout_marginLeft="30dp"
    android:layout_marginTop="48dp"
    android:text="Попыток:"
    android:visibility="invisible"
    android:textAppearance="?android:attr/textAppearanceMedium" />

<Button
    android:id="@+id/button_login"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/edit_password"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="94dp"
    android:onClick="Login"
    android:text="Войти" />
```

```
<TextView
    android:id="@+id/number_of_attempts"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignRight="@+id/user_text"
    android:layout_alignTop="@+id/attempts"
    android:visibility="invisible" />

<TextView
    android:id="@+id/login_locked"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:visibility="invisible"
    android:layout_alignParentBottom="true"
    android:layout_marginBottom="57dp"
    android:layout_alignRight="@+id/edit_password"
    android:layout_alignEnd="@+id/edit_password"
    android:layout_alignLeft="@+id/attempts"
    android:layout_alignStart="@+id/attempts" />


</android.support.constraint.ConstraintLayout>
```